MEMORANDUM
RM-4162-PR
JUNE 1964

JOSS: SCENARIO OF A FILMED REPORT

C. L. Baker

PREPARED FOR:
UNITED STATES AIR FORCE PROJECT RAND

The RAND Corporation
SANTA MONICA • CALIFORNIA

MEMORANDUM
RM-4162-PR
JUNE 1964

JOSS: SCENARIO OF A FILMED REPORT

C. L. Baker

This research is sponsored by the United States Air Force under Project RAND—Contract No. AF 49(638)-700 monitored by the Directorate of Development Plans, Deputy Chief of Staff, Research and Development, Hq USAF. Views or conclusions contained in this Memorandum should not be interpreted as representing the official opinion or policy of the United States Air Force.

DDC AVAILABILITY NOTICE
Qualified requesters may obtain copies of this report from the Defense Documentation Center (DDC).

The RAND Corporation
1700 MAIN ST • SANTA MONICA • CALIFORNIA • 90406

PREFACE

This Memorandum is a somewhat modified scenario of a 22-minute film, "JOSS," made at The RAND Corporation in March 1964, demonstrating the use of an on-line, time-shared computer system. "JOSS" stands for "JOHNNIAC Open Shop System."

The goal of the JOSS research program has been to provide a personal computing service specifically tailored to the needs of individual members of RAND's technical staff, rather than to simply provide a remote computer console communicating with an existing computer operating system. The emphasis has been on the development of a tool for problem solving rather than production computing; this implies, of course, intimate interaction between man and machine.

With this goal in mind, it was felt imperative that both the hardware (i.e., the remote typewriter consoles) and the software (i.e., the language and its use) be designed specifically for the intended application, as opposed to adapting an existing language and terminals to the proposed use. Speed and power, where necessary, have been sacrificed to the requirement for smooth operation and a natural language.

The demonstration film, produced for RAND by Sigma Educational Films, is intended as an introduction only, and although most of the features of the system are demonstrated, it has not been possible to show the rapid interaction which constitutes the typical mode of operation that makes JOSS so useful in problem solving. Although the film

is intended primarily for computer-oriented audiences, in a number of showings to lay audiences it has been found that it is readily understood by anyone with a knowledge of basic mathematics; this is due to the ease with which the user converses with JOSS.

Supplementing the information presented in the film, Appendix B contains a somewhat edited transcript of the discussion period which followed a showing of the film at International Business Machines Corporation, Poughkeepsie, New York, on April 29, 1964.

Answers to many of the most frequently-asked questions about JOSS may be found here.

A limited number of prints of the film are available for loan on a short-term basis. Inquiries should be directed to: The RAND Corporation, JOSS Film Distribution, 1700 Main Street, Santa Monica, California 90406.

SUMMARY

JOSS (JOHNNIAC Open Shop System) is an on-line, time-shared application of the RAND JOHNNIAC computer,† providing computational service at a number of remote electric typewriter consoles. As many as eight users at a time may communicate with JOSS by means of a smooth language, specifically designed for on-line problem solving.

JOSS handles a wide range of formula-evaluation type problems, providing the facility of a printing desk calculator as well as the capabilities of a stored-program computer. JOSS provides various arithmetic and trigonometric functions, and allows flexible output in user-specified formats. At the present time, the eight consoles within the RAND building--connected to a special-purpose input/output buffer by telephone lines--include semi-private stations for selected RAND researchers, and public stations for use by staff members on a first-come-first-served basis.

A console consists of a standard IBM Model 868 electric typewriter with a slightly modified character set and a small box with a few indicator lights and activating switches.

---

†The JOHNNIAC computer was designed in 1950-51 at The RAND Corporation, based on the Princeton Machine model; it has 4096 40-bit words of core memory, with an add time of 50 μsec. The only additional storage consists of a 12,288-word drum store, arranged in three 4096-word sections. The drum heads move to access a new section--this motion requires some 300 msec. The internal order code is austere--no indexing, no indirect addressing, no floating point, no interrupt mechanism. Part of JOHNNIAC is transistorized, but it is still largely composed of vacuum tube circuitry.

It uses continuous-feed, sprocket-guide, single-ply paper in 8½"×11" sheets. The user's input of instructions and data is typed in green, and JOSS responds with output in black.

This Memorandum contains the narration accompanying the filmed demonstration, the user's inputs, and the resulting computer outputs; only a minimum amount of the "staging" and "action" is described. Appendix A contains statements of the problems which were solved during the demonstration and the longer, more detailed outputs, some of which are shown only briefly in the film.

Appendix B presents answers to frequently-asked questions about the system.

CONTENTS

JOSS Console and Station Local Control Box

JOSS: Scenario of a Filmed Report

The JOSS film is divided into three parts. During the introductory scenes, J. C. Shaw, JOSS designer,[†] summarizes the purposes of the film and briefly shows a JOSS console station and the JOHNNIAC computer. He then introduces F. J. Gruenberger and C. L. Baker who demonstrate the use of JOSS. Willis H. Ware, Head, Computer Sciences Department, closes the report with some comments on the future of systems such as JOSS and the directions in which next-generation research is likely to develop.

In the scenario, these abbreviations are used: JCS--J. C. Shaw; U--the typed inputs of the JOSS user (F. J. Gruenberger); J--the typed outputs from JOSS; N--commentary of the narrator (C. L. Baker); WHW--Willis H. Ware. In the film, the U and J actions are, of course, typed out on the JOSS typewriter console.

JCS: This is a report on a research project conducted at The RAND Corporation under Air Force sponsorship. The aim has been to furnish the individual scientist with a modest computational capability on a demand basis. We call the system JOSS--standing for JOHNNIAC Open Shop System. It is an on-line, time-shared use of the computer.

JOHNNIAC is a venerable machine, but small and slow by today's standards. Coupled to it is a special-purpose buffer for messages to and from remote consoles. Each console consists of an electric typewriter and a few buttons and lights. At present, eight such stations in various locations in the RAND building are connected by telephone lines to the buffer.

[†]The multiple typewriter communication system used in JOSS was designed by T. O. Ellis and M. R. Davis.

[User presses POWER button, and JOSS responds by typing:]

J: Press 'IN' for JOSS service.

[User presses IN button.]

J: Please identify yourself.
U: RAND
J: 3-21-64 RAND†
U: Type 2+2.
J: 2+2 = 4

JCS: An executive routine in JOHNNIAC serves up to eight users at a time by interpreting and carrying out commands given in a simple language. Let's watch Fred Gruenberger and Charlie Baker demonstrate some of the features of JOSS.

N: JOSS may be used in three ways: as a printing desk calculator; as a stored-program computer; and, most productively, as a combination of these two. First, we will demonstrate some of JOSS' arithmetic capabilities.

U: Type 2+2-3-3+4+4.
J: 2+2-3-3+4+4 = 6

N: Multiplication must be indicated by a centered dot.

U: Type -1.2·1.2.
J: -1.2·1.2 = -1.44

N: We divide by using a slash,...

U: Type 7/2.
J: 7/2 = 3.5

N: ...and an attempt to divide by zero will result in an error message.

---

†This line is typed by JOSS at the top of a new page ($8\frac{1}{2}$"×11"). One-inch margins and similar date-lines are automatically supplied for each page.

```
U:    Type 7/0.
J:    Error above:  Zero divisor.

N:    An exponent is indicated by an asterisk.

U:    Type 2*8.
J:    2*8 = 256

N:    Several expressions may be evaluated on one line,
      and a blank line is indicated by an underscore.

U:    Type 3+2,3-2,3·2,_,3/2,3*2.
J:    3+2 = 5
      3-2 = 1
      3·2 = 6

      3/2 = 1.5
      3*2 = 9

N:    The arithmetic expression shown in this chart [see
      Chart A in Appendix A] may be evaluated by the
      addition of parentheses and brackets.

U:    Type [.7*5-2.3*(-.7)]/[7*6-|9*5-5*7|*(2/3)].
J:    [.7*5-2.3*(-.7)]/[7*6-|9*5-5*7|*(2/3)] = -3.33630357·10*(-6)

N:    A number of functions are available.  The argument is
      enclosed in parentheses.

U:    Type sqrt(49).
J:    sqrt(49) = 7

U:    Type sqrt(sqrt(sqrt(sqrt(3*2*2*2*2)))).
J:    sqrt(sqrt(sqrt(sqrt(3*2*2*2*2)))) = 3

N:    Trigonometric functions are provided, and we notice
      that numbers may be up to nine decimal digits.

U:    Type cos (3.14159265/4).
J:    cos(3.14159265/4) = .707106781

N:    Logarithms are to the base e.

U:    Type log(2.71828183).
J:    log(2.71828183) = 1
```

```
U:    Type y.
J:    Error above:  Undefined.

N:    JOSS is correct.  The variable y is undefined.  But
      a "Set" statement will serve to define it.

U:    Set y = exp(10).
      Type y.
J:    y = 22026.4658

N:    JOSS functions are available to take the integer part
      of a number, the fraction part, the digit part, and
      the exponent part.

U:    Type y, ip(y), fp(y), dp(y), xp(y).
J:        y = 22026.4658
      ip(y) = 22026
      fp(y) =      .4658
      dp(y) =     2.20264658
      xp(y) =     4

N:    To use JOSS as a stored-program computer, we may ask
      JOSS to store a statement rather than execute im-
      mediately.  This is indicated by prefixing the state-
      ment with an identifying number.

U:    1.1 Type x, sqrt(x), log(x), exp(x), _.

N:    We will now ask JOSS to execute this step a number of
      times, varying x from 1, in steps of 1, to 100.

U:    Do step 1.1 for x = 1(1)100.
J:          x =  1
      sqrt(x) =  1
       log(x) =  0
       exp(x) =  2.71828183

            x =  2
      sqrt(x) =  1.41421356
       log(x) =   .69314718
       exp(x) =  7.3890561

N:    Our table would look much better with each set of
      values on one line.  Let's interrupt JOSS by pressing
      the INTERRUPT button.  [User presses INTERRUPT button.]
      JOSS has calculated several lines and will stop when
      they have been printed.
```

J:
```
      x =  3
sqrt(x) =  1.73205081
 log(x) =  1.09861229
 exp(x) = 20.0855369

      x =  4
Interrupted at step 1.1.
```

N: We will change our program step by replacing it with one that calls for typing in a form.

U: 1.1 Type x, sqrt(x), log(x), exp(x), in form 1.

N: We must define this form.

U: Form1:*

N: JOSS does not require us to be perfect typists. A line may be killed by ending it with an asterisk, and JOSS will ignore this line.

U: Form 1:
```
___  __._______  __._______  ............
```

N: We have corrected our mistake; we may now use underscores in the form to indicate the position of fixed point numbers, and a string of dots to indicate that scientific notation is desired.

U: Go.

N: "Go" tells JOSS to resume calculation at the point of interruption.

J:
```
  4   2.0000000   1.3862944   5.4598150 01
  5   2.2360680   1.6094379   1.4841316 02
  6   2.4494897   1.7917595   4.0342879 02
  7   2.6457513   1.9459102   1.0966332 03
  8   2.8284271   2.0794415   2.9809580 03
  9   3.0000000   2.1972246   8.1030839 03
Interrupted at step 1.1.
```

N: Our output looks much better now. Let's dress it up with a heading. Again, we have interrupted.

U: 1.05 Do part 2 if fp(x/40) = 1/40.

N: JOSS will insert step 1.05 before our "Type" statement, and will request part 2 as a subroutine every forty lines.

U: 2.1 Page.
2.2 Type form 2.

N: Part 2--that is, all steps numbered two-point-something--asks for a new page and a heading line. We must define this heading.

U: Form 2:
X SQRT(X) LOG(X) EXP(X)
Do part 1 for x = 1(1)100.

N: This time we have asked JOSS to start all over from the beginning.

J:

X	SQRT(X)	LOG(X)	EXP(X)
1	1.0000000	.0000000	2.7182818 00
2	1.4142136	.6931472	7.3890561 00
3	1.7320508	1.0986123	2.0085537 01
4	2.0000000	1.3862944	5.4598150 01
5	2.2360680	1.6094379	1.4841316 02
6	2.4494897	1.7917595	4.0342879 02
7	2.6457513	1.9459102	1.0966332 03
8	2.8284271	2.0794415	2.9809580 03

Interrupted at step 1.05.

N: This looks much better, but an occasional space between lines would dress up our output nicely, so we have interrupted again. Step 1.07 should give us an extra space every five lines.

U: 1.07 Line if fp(x/5) = 1/5.
Do part 1 for x = 1(1)100.

[See Example 1 in Appendix A for JOSS' output at this point.]

N: Now that our program seems to be checked out, we may ask JOSS for a copy of the program, the forms, and the value of all variables.

U: Type all.
J: 1.05 Do part 2 if fp(x/40) = 1/40.
1.07 Line if fp(x/5) = 1/5.
1.1 Type x, sqrt(x), log(x), exp(x) in form 1.

2.1 Page.
2.2 Type form 2.

Form 1:

—— —·——— —·——— .....

Form 2:
X SQRT(X) LOG(X) EXP(X)

x = 100
y = 22026.4658

N: We can now communicate with JOSS well enough to attempt a problem. An example is found in McCracken's Guide to FORTRAN Programming† [see Chart B in Appendix A]. First, we'll need the constant 2π.

U: Set k = 2·3.14159265.

N: Next, we can store a program to compute the required factors, and an approximate time interval [see Chart C]...

U: 1.1 Set F = sqrt[1/(L·C)]/k.
1.2 Set f = sqrt[1/(L·C)-R*2/(4·L*2)]/k.
1.3 Set I = k·F*2·Q/f.
1.4 Set d = .1/f.

N: ...and we'll type out these values.

U: 1.5 Type F, f, I, d.

N: Next, we need to enter the parameters for our specific problem.

---

†McCracken, D. D., A Guide to FORTRAN Programming, John Wiley and Sons, Inc., New York, 1961.

U: Set R = 100.
Set L = .2.
Set C = .5·10*(-6).
Set Q = .001.

N: Now we ask JOSS to compute these factors.

U: Do part 1.
J: F = 503.292122
f = 501.716868
I = 3.17220634
d = 1.99315603·10*(-4)

N: These look fine, with the exception of that rather messy looking time interval. We can change it to a clean one, and, knowing that our interval is small, our program can print the time in milliseconds rather than seconds.

U: Set d = .0002.
2.1 Type 1000·t, I·exp[-R·t/(2·L)] sin(k·f·t) in form 3.
Form 3:
__.__ ms. -_._____ amp.

N: The program to compute and type consists of a single step which will print in a single form. The form may include any additional information we wish to print out with our answers. We can now ask JOSS to execute our step for 100 time intervals, from t = 0, in steps of d, to 100 times d.

U: Do part 2 for t = 0(d)100·d.

[See Example 2 in Appendix A for JOSS' output at this point.]

N: Since our program seems to be gold star, let's record it.

U: Type all.

[See Example 3 in Appendix A for JOSS' output at this point.]

N: The program is finished, so we might as well erase it, its forms, and all associated values from JOSS' storage.

U: Delete all.

N: That was admittedly a textbook problem. In preparing its work for the Air Force, RAND authors frequently have the problem of evaluating an expression for a number of values to aid in the preparation of a graph. Here is one such expression [see Chart D in Appendix A]. Let's do it on JOSS. We'll start on a page with a heading.

U:
```
1.1 Page.
1.2 Type form 1.
Form 1:
 P      L       M
1.3 Do part 2 for p = 81, 72, 45, 9.
2.1 Line.
```

N: Step 1.3 will evaluate our formula for four values of p--81, 72, 45, and 9;...

U: 2.2 Do part 3 for L = .1(.2(.2)2.

N: ...and for L equals .1--here we can backspace, strike over the error,† and go on--for L equals .1, .2 in steps of .2 up to 2.

U:
```
3.1 Set m = -2.5·k·log[cos(p/c)·.64/sqrt(.5)/L*2].
3.2 Type p, L, m in form 2.
Form 2:
```

N: Part 3 evaluates our expression and types out the answers. Next, we need to enter the constants required to convert degrees to radians, and to convert logs from base e to base 10. Then we can ask JOSS for the results.

---

†Note that the user inadvertently struck the left-paren key; the error is rectified by simply backspacing and striking-over with the intended character, the comma.

U: Set c = 180/3.14159265.
Set k = 1/log(10).
Do part 1.

[See Example 4 in Appendix A for JOSS' output at this point.]

N: Again, we'll keep a record of our program before we delete it.

U: Type all.

[See Example 5 in Appendix A for JOSS' output at this point.]

U: Delete all.

N: JOSS can operate with indexed variables, and a stored program may call for values of variables to be entered during execution.

U: 1.1 Demand q(x,y).

N: We'll vary the value of the first index x from 1, in steps of 1, to y,...

U: 2.1 Do part 1 for x = 1(1)y.

N: ...and vary the index y from 1, in steps of 1, to 8.

U: Do part 2 for y = 1(1)8.

N: We enter the value of each variable on the same line as it is requested by JOSS.

J:	q(1,1) =	U:	3.456
	q(1,2) =		-4.32
	q(2,2) =		8
	q(1,3) =		4.324567
	q(2,3) =		-56.45
	q(3,3) =		0.000987
	q(1,4) =		4.44444444
	q(2,4) =		-4.444433
	q(3,4) =		

N: When all these values have been entered, we may proceed to have JOSS calculate with indexed variables as would be required for problems more complex than we have been able to demonstrate here.

WHW: You have just seen a demonstration of many features of the JOSS system. As was pointed out, this is an experimental system designed to provide individual scientists and engineers with a personal, on-demand computing service. As a side aspect of the experiment, we hope to gain insight into the interface problem between a man and a computer.

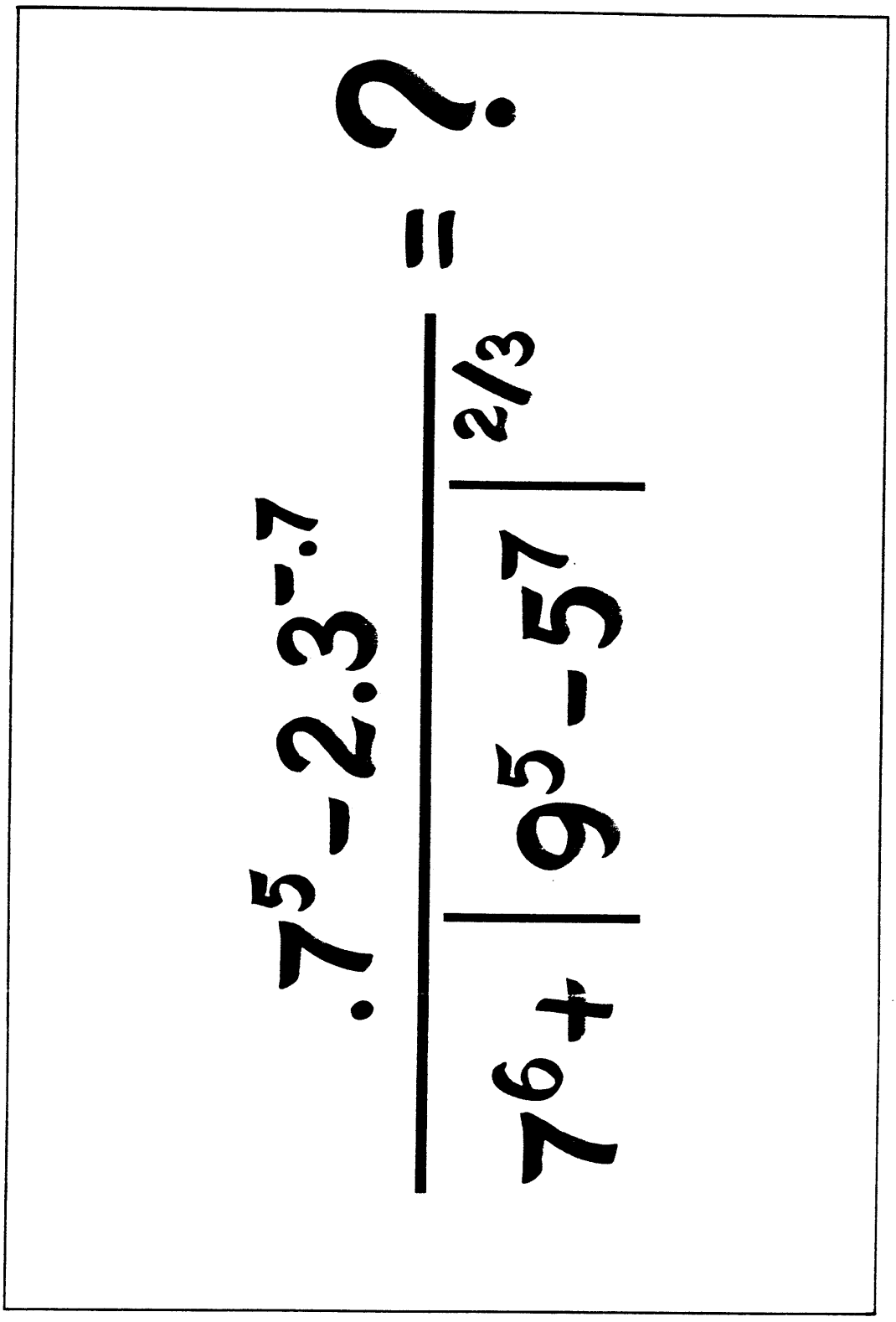
The system shown here is currently available to RAND's professional staff. Our experience in using JOSS, our observations of user behavior, will be the basis on which to continue research. Later-model systems will certainly differ from what you have seen.

JOSS is one of many current efforts in on-line time-shared use of computers. The goal of the JOSS project has been to demonstrate that the ability to be on-line with the computer, given a reasonable language and almost continuous interaction, leads to a powerful computational tool. It is probably too early to forecast the ultimate effect of such systems, but for a certain class of problems, at least, the programmer as the middleman between the problem and the machine is no longer needed.

JOSS represents, we believe, a significant step forward because of its intimate interaction between man and machine.

Appendix A

Chart A: CHARTS AND EXAMPLES



Example 1:

X	SQRT(X)	LOG(X)	EXP(X)
1	1.0000000	.0000000	2.7182818 00
2	1.4142136	.6931472	7.3890561 00
3	1.7320508	1.0986123	2.0085537 01
4	2.0000000	1.3862944	5.4598150 01
5	2.2360680	1.6094379	1.4841316 02
6	2.4494897	1.7917595	4.0342879 02
7	2.6457513	1.9459102	1.0966332 03
8	2.8284271	2.0794415	2.9809580 03
9	3.0000000	2.1972246	8.1030839 03
10	3.1622777	2.3025851	2.2026466 04
11	3.3166248	2.3978953	5.9874142 04
12	3.4641016	2.4849067	1.6275479 05
13	3.6055513	2.5649494	4.4241339 05
14	3.7416574	2.6390573	1.2026043 06
15	3.8729834	2.7080502	3.2690174 06
16	4.0000000	2.7725887	8.8861105 06
17	4.1231056	2.8332133	2.4154953 07
18	4.2426407	2.8903718	6.5659969 07
19	4.3588989	2.9444390	1.7848230 08
20	4.4721360	2.9957323	4.8516520 08
21	4.5825757	3.0445224	1.3188157 09
22	4.6904158	3.0910425	3.5849129 09
23	4.7958315	3.1354942	9.7448035 09
24	4.8989795	3.1780538	2.6489122 10
25	5.0000000	3.2188758	7.2004899 10
26	5.0990195	3.2580965	1.9572961 11
27	5.1961524	3.2958369	5.3204824 11
28	5.2915026	3.3322045	1.4462571 12
29	5.3851648	3.3672958	3.9313343 12
30	5.4772256	3.4011974	1.0686475 13
31	5.5677644	3.4339872	2.9048850 13
32	5.6568543	3.4657359	7.8962960 13
33	5.7445627	3.4965076	2.1464358 14
34	5.8309519	3.5263605	5.8346174 14
35	5.9160798	3.5553481	1.5860135 15
36	6.0000000	3.5835189	4.3112316 15
37	6.0827625	3.6109179	1.1719142 16
38	6.1644140	3.6375862	3.1855932 16
39	6.2449980	3.6635617	8.6593400 16
40	6.3245553	3.6888795	2.3538527 17

X	SQRT(X)	LOG(X)	EXP(X)
41	6.4031242	3.7135721	6.3984349 17
42	6.4807407	3.7376696	1.7392749 18
43	6.5574385	3.7612001	4.7278395 18
44	6.6332496	3.7841896	1.2851600 19
45	6.7082039	3.8066625	3.4934271 19
46	6.7823300	3.8286414	9.4961194 19
47	6.8556546	3.8501476	2.5813129 20
48	6.9282032	3.8712010	7.0167359 20
49	7.0000000	3.8918203	1.9073466 21
50	7.0710678	3.9120230	5.1847055 21
51	7.1414284	3.9318256	1.4093491 22
52	7.2111026	3.9512437	3.8310080 22
53	7.2801099	3.9702919	1.0413759 23
54	7.3484692	3.9889841	2.8307533 23
55	7.4161985	4.0073332	7.6947853 23
56	7.4833148	4.0253517	2.0916595 24
57	7.5498344	4.0430513	5.6857200 24
58	7.6157731	4.0604430	1.5455389 25
59	7.6811458	4.0775374	4.2012104 25
60	7.7459667	4.0943446	1.1420074 26
61	7.8102497	4.1108739	3.1042979 26
62	7.8740079	4.1271344	8.4383567 26
63	7.9372539	4.1431347	2.2937832 27
64	8.0000000	4.1588831	6.2351491 27
65	8.0622578	4.1743873	1.6948892 28
66	8.1240384	4.1896547	4.6071866 28
67	8.1853528	4.2046926	1.2523632 29
68	8.2462113	4.2195077	3.4042761 29
69	8.3066239	4.2341065	9.2537817 29
70	8.3666003	4.2484952	2.5154387 30
71	8.4261498	4.2626799	6.8376712 30
72	8.4852814	4.2766661	1.8586717 31
73	8.5440038	4.2904594	5.0523936 31
74	8.6023253	4.3040651	1.3733830 32
75	8.6602540	4.3174881	3.7332420 32
76	8.7177979	4.3307333	1.0148004 33
77	8.7749644	4.3438054	2.7585135 33
78	8.8317609	4.3567088	7.4984170 33
79	8.8881944	4.3694479	2.0382811 34
80	8.9442719	4.3820266	5.5406224 34

X	SQRT(X)	LOG(X)	EXP(X)
81	9.0000000	4.3944492	1.5060973 35
82	9.0553851	4.4067193	4.0939970 35
83	9.1104336	4.4188406	1.1128638 36
84	9.1651514	4.4308168	3.0250773 36
85	9.2195445	4.4426513	8.2230127 36
86	9.2736185	4.4543473	2.2352466 37
87	9.3273791	4.4659081	6.0760302 37
88	9.3808315	4.4773368	1.6516363 38
89	9.4339811	4.4886364	4.4896128 38
90	9.4868330	4.4998097	1.2204033 39
91	9.5393920	4.5108595	3.3174001 39
92	9.5916631	4.5217886	9.0176284 39
93	9.6436508	4.5325995	2.4512455 40
94	9.6953597	4.5432948	6.6631762 40
95	9.7467943	4.5538769	1.8112391 41
96	9.7979590	4.5643482	4.9234583 41
97	9.8488578	4.5747110	1.3383347 42
98	9.8994949	4.5849675	3.6379710 42
99	9.9498744	4.5951199	9.8890303 42
100	10.0000000	4.6051702	2.6881171 43

Chart B:

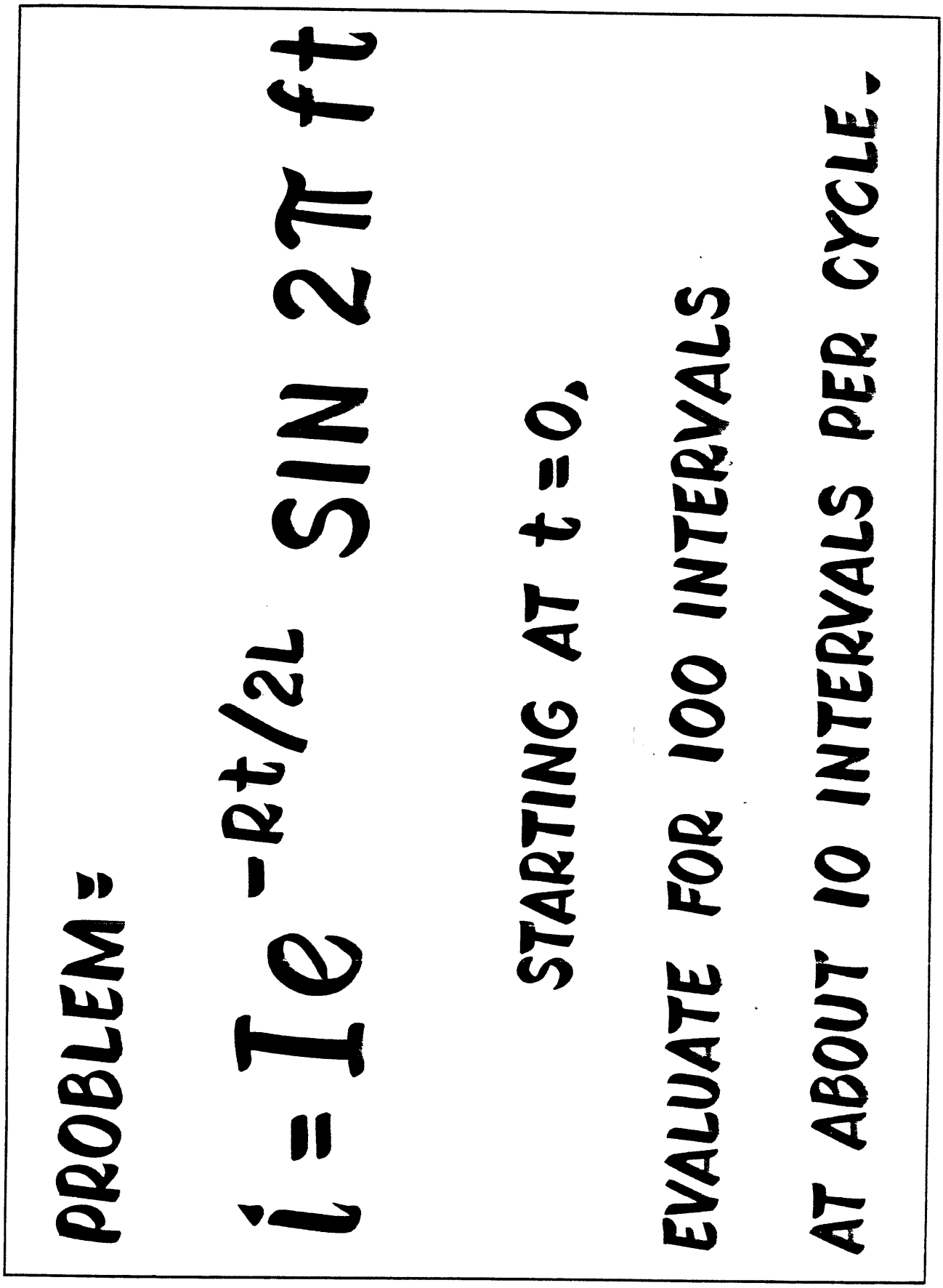

PROBLEM =
$i = Ie^{-Rt/2L} \sin 2\pi ft$
STARTING AT $t = 0$,
EVALUATE FOR 100 INTERVALS
AT ABOUT 10 INTERVALS PER CYCLE.

Chart C:

PARAMETERS REQUIRED: R, L, C, AND Q

CONSTANTS TO BE CALCULATED

$$F = \frac{1}{2\pi}\sqrt{\frac{1}{LC}}$$

$$f = \frac{1}{2\pi}\sqrt{\frac{1}{LC} - \frac{R^2}{4L^2}}$$

$$I = \frac{2\pi F^2 Q}{f}$$

INTERVAL $\approx .1/f$

Example 2:

.00 ms.	.000000 amp.
.20 ms.	1.778902 amp.
.40 ms.	2.733649 amp.
.60 ms.	2.591195 amp.
.80 ms.	1.508396 amp.
1.00 ms.	-.026650 amp.
1.20 ms.	-1.405807 amp.
1.40 ms.	-2.136197 amp.
1.60 ms.	-2.010678 amp.
1.80 ms.	-1.156910 amp.
2.00 ms.	.041508 amp.
2.20 ms.	1.110600 amp.
2.40 ms.	1.669108 amp.
2.60 ms.	1.560014 amp.
2.80 ms.	.887011 amp.
3.00 ms.	-.048484 amp.
3.20 ms.	-.877107 amp.
3.40 ms.	-1.303985 amp.
3.60 ms.	-1.210201 amp.
3.80 ms.	-.679828 amp.
4.00 ms.	.050339 amp.
4.20 ms.	.692491 amp.
4.40 ms.	1.018606 amp.
4.60 ms.	.938705 amp.
4.80 ms.	.520840 amp.
5.00 ms.	-.048997 amp.
5.20 ms.	-.546570 amp.
5.40 ms.	-.795583 amp.
5.60 ms.	-.728019 amp.
5.80 ms.	-.398878 amp.
6.00 ms.	.045781 amp.
6.20 ms.	.431272 amp.
6.40 ms.	.621313 amp.
6.60 ms.	.564544 amp.
6.80 ms.	.305351 amp.
7.00 ms.	-.041586 amp.
7.20 ms.	-.340199 amp.
7.40 ms.	-.485157 amp.
7.60 ms.	-.437718 amp.
7.80 ms.	-.233656 amp.
8.00 ms.	.037003 amp.
8.20 ms.	.268284 amp.
8.40 ms.	.378791 amp.
8.60 ms.	.339337 amp.
8.80 ms.	.178717 amp.
9.00 ms.	-.032410 amp.
9.20 ms.	-.211514 amp.
9.40 ms.	-.295709 amp.
9.60 ms.	-.263032 amp.
9.80 ms.	-.136634 amp.
10.00 ms.	.028035 amp.
10.20 ms.	.166713 amp.
10.40 ms.	.230822 amp.
10.60 ms.	.203857 amp.

10.80 ms.	.104412 amp.
11.00 ms.	-.024007 amp.
11.20 ms.	-.131368 amp.
11.40 ms.	-.180151 amp.
11.60 ms.	-.157972 amp.
11.80 ms.	-.079750 amp.
12.00 ms.	.020387 amp.
12.20 ms.	.103490 amp.
12.40 ms.	.140586 amp.
12.60 ms.	.122398 amp.
12.80 ms.	.060882 amp.
13.00 ms.	-.017193 amp.
13.20 ms.	-.081508 amp.
13.40 ms.	-.109698 amp.
13.60 ms.	-.094821 amp.
13.80 ms.	-.046454 amp.
14.00 ms.	.014412 amp.
14.20 ms.	.064180 amp.
14.40 ms.	.085586 amp.
14.60 ms.	.073447 amp.
14.80 ms.	.035426 amp.
15.00 ms.	-.012019 amp.
15.20 ms.	-.050524 amp.
15.40 ms.	-.066765 amp.
15.60 ms.	-.056883 amp.
15.80 ms.	-.027000 amp.
16.00 ms.	.009978 amp.
16.20 ms.	.039765 amp.
16.40 ms.	.052078 amp.
16.60 ms.	.044047 amp.
16.80 ms.	.020566 amp.
17.00 ms.	-.008252 amp.
17.20 ms.	-.031289 amp.
17.40 ms.	-.040616 amp.
17.60 ms.	-.034103 amp.
17.80 ms.	-.015656 amp.
18.00 ms.	.006800 amp.
18.20 ms.	.024615 amp.
18.40 ms.	.031673 amp.
18.60 ms.	.026400 amp.
18.80 ms.	.011910 amp.
19.00 ms.	-.005586 amp.
19.20 ms.	-.019360 amp.
19.40 ms.	-.024697 amp.
19.60 ms.	-.020434 amp.
19.80 ms.	-.009054 amp.
20.00 ms.	.004576 amp.

Example 3:

```
1.1 Set F = sqrt[1/(L·C)]/k.
1.2 Set f = sqrt[1/(L·C)-R*2/(4·L*2)]/k.
1.3 Set I = k·F*2·Q/f.
1.4 Set d = .1/f.
1.5 Type F, f, I, d.

2.1 Type 1000·t, I·exp[-R·t/(2·L)]·sin(k·f·t) in form 3.


Form 3:
__.__ ms.    -_._____ amp.


        d =      2·10*(-4)
        f =    501.716868
        k =      6.2831853
        t =      2·10*(-2)
        C =      5·10*(-7)
        F =    503.292122
        I =      3.17220634
        L =       .2
        Q =      1·10*(-3)
        R =    100
```

Chart D:

We can now calculate the difference in brightness from the tabular entry as ℓ_2 and P change, holding ℓ_3 constant. This difference, Δm, is given by

$$\Delta m = -2.5 \log \left(\frac{\cos P}{0.707} \cdot \frac{0.64}{\ell_2^2} \right).$$

[Source: Kocher, G. E., and D. T. Jamison, A Deep-Space Triangulation Probe to Determine the Astronomical Unit, The RAND Corporation, RM-4014-NASA, January 1964, p. 47.]

Example 4:

P	L	M
81	.10	-2.88
81	.20	-1.37
81	.40	.13
81	.60	1.01
81	.80	1.64
81	1.00	2.12
81	1.20	2.52
81	1.40	2.85
81	1.60	3.14
81	1.80	3.40
81	2.00	3.63
72	.10	-3.62
72	.20	-2.11
72	.40	-.61
72	.60	.27
72	.80	.90
72	1.00	1.38
72	1.20	1.78
72	1.40	2.11
72	1.60	2.40
72	1.80	2.66
72	2.00	2.89
45	.10	-4.52
45	.20	-3.01
45	.40	-1.51
45	.60	-.62
45	.80	.00
45	1.00	.48
45	1.20	.88
45	1.40	1.22
45	1.60	1.51
45	1.80	1.76
45	2.00	1.99
9	.10	-4.88
9	.20	-3.37
9	.40	-1.87
9	.60	-.99
9	.80	-.36
9	1.00	.12
9	1.20	.52
9	1.40	.85
9	1.60	1.14
9	1.80	1.40
9	2.00	1.63

Example 5:

```
1.1 Page.
1.2 Type form 1.
1.3 Do part 2 for p = 81, 72, 45, 9.

2.1 Line.
2.2 Do part 3 for L = .1, .2(.2)2.

3.1 Set m = -2.5·k·log[cos(p/c)·.64/sqrt(.5)/L*2].
3.2 Type p, L, m in form 2.


Form 1:
 P       L        M
Form 2:
—    - —    -_·__


        c =      57.2957796
        k =        .434294482
        m =       1.62686274
        p =       9
        L =       2
```

Appendix B

JOSS FILM QUESTIONS AND ANSWERS

This appendix, supplementing the information presented in the film, contains a somewhat edited transcript of the discussion period which followed a showing of the film at International Business Machines Corporation, Poughkeepsie, New York, on April 29, 1964. Answers to many of the most frequently-asked questions about JOSS may be found here. The question-and-answer session was conducted by C. L. Baker.

1) Q: IS THE SYSTEM BEING USED?

A: Oh, boy, is it. Our operating hours at present are from 11 am to 3:30 pm, and from 5 pm on, and we're continually pressed by users to expand the schedule.

2) Q: AT THE PRESENT TIME, DO YOU ALLOW ONLY ONE USER ON THE SYSTEM AT A TIME?

A: No; the system is time-shared among eight users. We have three private consoles, which are given to scientists who we feel will make good use of the system. Actually, our goal was to give each user a console of his own that he would treat much as he treats his telephone. It's at his desk, he can use it as he wants, use it when he wants--and when he goes on vacation, no one will come in and take it out because it's sitting there idle.

In addition to these three private stations, we have five public stations available to any member of the RAND staff. To enable users to determine when a public console is available, we have installed a dial-up telephone system. At the first telephone number, a busy signal tells you that the JOSS system is in operation. At other numbers, a busy

signal tells you that the corresponding console is in operation, so you don't have to go trotting down the halls trying to find an empty console. In practice, however, the public stations are in use for long periods of time, so at each console there's a sign-up sheet; when you are done, you phone the next person on the list.

3) Q: WHO ARE THE MAIN USERS OF THE SYSTEM?

A: The professional programmer in our closed-shop operation makes very little use of it except to get answers to test cases, and things like that. The open-shopper who has FORTRAN experience likes JOSS very much, but has difficulty using the system. He tends to do things in the FORTRAN way, which is not well-adapted to this on-line conversational mode. Our greatest response has been from those engineers and scientists within the building who have tried, unsuccessfully, to come in contact with computers before, and have gotten hung up on the strange language or the artificial barriers of the required mechanics. These people are throwing away their slide rules as quickly as they can get to a JOSS console.

As you noticed in the very first part of the film, we showed the complete turn-on sequence for the machine. We walked up to a dead machine and turned on the power (the power lets it be used also as an electric typewriter, of course). We turned on the system, the system identified itself and asked us to identify ourselves, so that each page could be headed with our little commercial on it. From that point on, we were in business. The simplest thing you can do takes 15 key strokes: turn on the system, do all initiation steps, and get an answer back--that's to the problem "Type 2+2."

The extreme simplicity of this turn-on-and-start-computing procedure has been a tremendously important factor in attracting many users to the JOSS system.

4) Q: IS IT BEING USED PRIMARILY AS A DESK CALCULATOR OR AS A STORED-PROGRAM MACHINE?

A: As a combination of the two. There is intimate interaction between the user and JOSS. In an actual problem-solving situation, the ribbon color changes from black to green almost every few lines. By problem-solving, we mean that the user is continually exercising judgment during the course of the computation--changing and modifying the procedure whenever he wishes. He starts something, sees how it goes, pushes the INTERRUPT button, changes it, tries again, changes the program, and so on. We showed only a couple of errors in the film. There are actually thirty error messages to which you must respond. You're completely safe, in the sense that anything you try to do that is illegal or unrecognizable to the system will result in an error message, but always in such a way that you can clean up and go on. There are no catastrophic errors at all.

5) Q: DO YOU HAVE ANY IDEA WHAT AMOUNT OF CONSUMED TIME IS INPUT/OUTPUT TIME?

A: The conversation time is high for the experienced user. The user of the system has sort of a double-plateau learning curve. The first plateau, reached in perhaps a few hours, is that of knowing the language. Very quickly you get to the point where you know what it can do. When a user has a question about the JOSS language, we encourage him to "try it and see what happens," which is a very effective teaching tool. At this plateau, you are using JOSS

much as you would use a desk calculator. The second plateau is reached at various levels according to one parameter only--the user's typing ability. This plateau is one place at which you begin to sort of put your arm around the thing and say, "Come on, old JOSS, let's do this together." And it's really that way. You start going back and forth, back and forth, conversing very, very quickly. It takes about a couple of weeks to learn this, and then only if you are a moderately skilled typist. This is one of the unfortunate parts of this system, in a way. On the other hand, we're rather proud of the statement that this is the biggest stumbling block to any user we have found, to date. His facility with the system is in direct proportion to his typing ability: the better the typing ability, the better he uses it; the poorer his typing ability, the more difficulty he has with it. But this is apparently a fairly easy skill to teach people early. And, as a matter of fact, JOSS has motivated at least one user to learn to touch-type, so that he could interact more easily.

6) Q: WHAT SORT OF USER TRAINING PROGRAM IS REQUIRED?

A: You have just seen it. However, most of the people who are using JOSS haven't had even this much formal training. Because we wanted to observe the initial response of a user, we tried very hard to keep preliminary JOSS usage to a minimum, and especially away from the people whose reactions we wanted to observe. While the system was being checked out, the word about JOSS spread through RAND much too quickly for us to be able to control the use in such a way as to make observation of novice-user response possible. I have a station outside my office and I can see who comes

to use it. Every day three or four new people come by. They work away for fifteen minutes, a half-hour, or an hour, and go away looking quite pleased. I don't know how they've learned, but there they are. We know that this language turns out to be very good for a conversational mode of operation. We found, too, that the large character set really helps, as do the capital and lower letters. It is much easier to use the language because sentences start with capitals and end with periods, and it's very easy to tell English commands from algebraic expressions. JOSS is a very readable language. We feel this is an extremely important characteristic for a language to have--a readable language is easy to learn.

We do have a list of the "okay words" available at each console. We plan to publish a series of examples of common things that you do all the time, such as solving for the zeros of functions, roots of polynomials, integration schemes, and a number of other common procedures.

7) Q: DO YOU THINK PEOPLE (ENGINEERS) ARE WASTING TIME OR LOSING EFFECTIVENESS BECAUSE THEY MUST PROGRAM THE MACHINE THEMSELVES?

A: Well, there are two ways of using JOSS because there are two groups of users. One is the engineer who is at the console himself. Ideally, he would have his own and wouldn't be under any pressure. However, the usual situation is that he's not alone, and he is under pressure, and perhaps he's not able to work most effectively this way. More consoles would take care of that. But if it's answer-getting rather than problem-solving, he would very likely turn this over to the second type of user: the computing aide, who can

program and run the job in whatever system is appropriate--from desk calculator on up. These computing aides tell us that JOSS has markedly increased their productivity.

8) Q: WHEN I VISITED YOU AT RAND ABOUT TWO WEEKS AGO, WE TALKED TO A USER AT THE STATION NEXT DOOR. HE WAS VERY ENTHUSIASTIC. PERHAPS YOU COULD SAY A FEW WORDS ABOUT HIS PROBLEM?

A: The problem that this fellow had was a Fourier transform problem. He wanted to determine the shape of a pulsed wave form after it had been passed through a filter. He knew the input pulse in the time domain, the filter response in the frequency domain, and decided that the frequency domain was the one in which to do his integration. So now he was confronted with a very complex integrand, which had to be integrated from zero to infinity many times in order to determine the output response. He didn't know what this output pulse was going to look like at this point, so he didn't know how many times he would have to integrate it. He didn't know the nature of the integrand, really, because it was a very complex algebraic function, but he had to do the job. He happened to show up at my console one day, and because I always like to observe user response to the system, we started working on the problem together. He was a brand new user of the system, but was making good progress when I sat down with him. Within two days we had his entire problem almost solved. By that time, we knew pretty much what the integrand looked like, and what form of numerical integration to use to take a minimum amount of time while giving sufficient accuracy. We were able to identify the errors resulting from numerical integration as opposed to those which arose from truncation of the

integrand at some finite point. We were able to tailor the integration scheme to the integrand and develop a program which would integrate with various step sizes over various length intervals, but always keeping aware of the maximum frequency within the system in order to insure stability. We eventually ended up relating the integration intervals and step sizes to the various parameters in the integrand. At this point, we really were masters of the functions involved in the problem. And, of course, he learned a lot of other things about his problem, I'm sure, such as how to synthesize different filters that would work a little bit better. But, as I'm sure you can imagine, in doing this with a brute-force method, he perhaps would have gotten answers, but he would not necessarily have known their reliability, he would have been in the dark about how to proceed on future problems, and, of course, the time span for doing this could have been several weeks or several months.

9) Q: DOES THE SYSTEM KEEP A LOG TO DETERMINE SUCH THINGS AS TIME ON THE SYSTEM, NUMBER OF USERS, LENGTH OF PROBLEMS, ETC.?

A: We wish we could. The system is a little bit too small for that. We would like to but we just have not been able to.

10) Q: DO YOU KNOW THE AVERAGE TIME A MAN IS AT A CONSOLE?

A: No, we don't. It's highly variable. I look outside my office and see people coming and going. Sometimes it's five minutes, sometimes it's 20, sometimes it's 3:00 in the morning before he leaves.

11) Q: HOW DO YOU KEEP THE RESPONSE TIME SMALL?

A: Our scheduling algorithm minimizes response time quite well with eight people on-line. The system runs

entirely interpretively, so that the supervisor (JOSS) is in control at all times. Now let's suppose that there are six or seven users on, which is the normal case, and that JOSS is interpreting the program of one of these users. At the end of each interpretation cycle, which is one statement within the machine, the supervisor will go and look at the I/O buffering system to see if, during the last interpretation cycle, anyone in the system has hit the carriage return--this is the signal to JOSS requesting processing of a line. It turns out that if there are six or seven people on-line, and all are in the conversational mode--putting in a line at a time and getting a response--you will notice as much as a second or second-and-a-half delay. Fortunately, it never goes over two seconds, and reaches that only if almost everybody hits the carriage return at the same time.

If, on the other hand, you happen to be in the execution mode, without printout, you don't expect an immediate response, and expect to wait for the computation to take place. At this point you may have at your disposal anything between an entire JOHNNIAC and one-eighth of a JOHNNIAC. You can therefore notice quite a variance in computation time--but not in initial response time.

12) Q: HOW DO YOU ALLOCATE COMPUTE TIME AMONG USERS?

A: It's on a clock basis, which means that at the end of every interpretation cycle you also look at the real-time clock. At the end of a user's allotted time slot, which is about two seconds, you go get the next user, and so forth. All compute-limited users go around and around.

13) Q: IS THIS TIME SLOT THE EQUIVALENT OF ONE LINE?

A: No. It's the equivalent of about 30 average interpretation cycles, meaning pick up a statement, scan it, execute it, and go on to the next one. At the end of each one of these cycles, JOSS looks at the clock, so it is dependent on how much time you have used rather than on the number of cycles. If everybody is computing, everybody will get a shot every sixteen seconds. That means your shots are about two seconds long.

14) Q: ARE PROGRAM STEPS COMPILED BEFORE BEING EXECUTED?

A: No. I'd like to stress the fact that this is an interpretive system. The internal representation of JOSS statements is identical to the external representation, as is the representation of decimal values.

We feel that this interpretive mode of operation is very important. Although "interpretation" has been a nasty word for many years, we don't think it should be, because an interpretive system can do a lot for you. It's hard for us to state simply exactly why interpretation is so useful, but the effect of interpretation is that the context in which your statement will be executed is determined at the time of execution and can change from execution to execution, giving you a very powerful language when you make use of this fact. For instance, anyplace where there is a number in the language, you may substitute a variable or an expression. For example, we might want to see a given form, say form 3, so we could say "Type form 3." We could also say "Type form x," and if the value of x happened to be 3, we would print form 3.

We could say, "Type form i" in a step, "Do" this step for i = 1 to 10, and print out forms 1 to 10. Similarly, we can print out steps individually, parts individually, and values individually. Any subscript may also be an expression, and the expression may involve subscripted variables whose subscripts are expressions--almost ad infinitum.

The same feature may be used in "for" statements. The limits of executing the "for" statement can be of the form "expression, in steps of expression, to an expression, in steps of another expression, to another expression," and so on as far as you want to go. This "limit-step-limit-etc." form may be one item on a list, and a list may have many items. Each of the components can be just about as complicated an expression as you want to make. All of this is, of course, finalized at the time of execution.

Another point that I should emphasize while we're on the subject is that the conditional clause doesn't just test the sign of a variable, but tests relationships between expressions. For instance, we can say: if a = b = c and y > z or x = y and relation R between expression U and expression V. This type of conditional statement may appear with any step or command, and it turns out that this is very, very powerful. In fact, in some programs all the computation is done in the conditional expressions; that is particularly true of the problem-solving type of work. And it reflects very naturally what is done in the textbooks on mathematics. You see on the left something that you do, and over on the right-hand side you see the conditions under which this holds. You can take a numerical analysis textbook and sort of copy it onto JOSS. In using this feature, it's

important for all calculations to be done in decimal, so that all "magic numbers" come out accurately. If you ask for a function you know, such as 2^{16} or sqrt(2), you'll get out the right number, exact to 9 decimal digits. If there is a fractional power, JOSS looks at the fraction, and if it turns out to be 1/2, you get the square root routine rather than the log and exponential routines. A lot of attention has been paid to making the numbers you think you know, come out the way you know them. As an example, the log of e is indeed 1, not .999999999 or 1.00000001 which can be pretty annoying to people.

15) Q: IS IT POSSIBLE FOR A STATION TO TIE THE SYSTEM IN A LOOP?

A: Absolutely impossible. The user can get a loop himself, of course, by asking for an endlessly repeated computation. The worst this can do is to degrade the system 1/8. The other people are going to get their time slots. Endless loops don't happen too often in practice, however, because rather than writing the loops yourself, you usually ask JOSS to do it as we did here. We didn't write any loops in the filmed examples; we let JOSS do the work.

16) Q: HOW DOES THE DIAGNOSTIC FUNCTION WORK?

A: Though it wasn't brought out explicitly, there are two types of commands. Those that are to be done immediately--such as "Do part so-and-so," or "Type y," etc.--are of course scanned and executed immediately and must be well formed with all variables defined at this point. Statements that are to be stored internally are not scanned until actual execution time, because execution time is the only time at which their context actually need exist. The only check that is

made on these is that there is a well-formed statement number preceding the statement, so that there is some place in the hierarchy of storage to put in and recall it properly. So there are two times: one at execution if it is internally stored, and one at direct entry time if it is to be executed directly. It is important to remember that when JOSS is interpreting steps in the internal mode and detects an error condition, it responds with an appropriate error message. The system then backs off so that you can correct the offending step or steps, change any of the variables, define new variables, put in a part if a part is missing, and so on. Then by saying "Go.", you ask JOSS to try again from the beginning of the execution of the statement which caused the error. As you noticed, the very first thing we did with a stored program was to start JOSS iterating over the step, interrupt, knock the whole step out from underneath the iteration control, put in a new step, and then start up again.

17) Q: WHAT ARE THE STORAGE LIMITATIONS FOR A PROGRAM?

A: Well, the primary limitation, of course, is JOHNNIAC, which as I said is a very small machine. Currently, each user--of which there are eight--has available for his own use a stock of JOSS cells. These JOSS cells are arranged in a list structure so that he uses these up and gives them back continually. Each JOSS cell holds nine characters. The internal representation is the same as the external representation in all cases, and there is currently a stock of 110 cells for each user, in addition to the cells which are needed for variables, control, and so on. This represents

about a page's worth of typing. We hope very shortly, when we get some of our drum transfer problems solved, to increase this to around 300-350.

18) Q: WHAT HAPPENS WHEN YOU RUN OUT OF SPACE?

A: When you override the limit, you come to an error message which will print out the words "Insufficient storage space," but there's enough safe space left to allow you to clean up. You can usually recover. For instance, if you are printing in a form, you may decide to eliminate the form and go back to JOSS-style output, which is much simpler and takes much less storage space. However, the problems are generally small. Our goal, again, was to get problem-solving interaction rather than production-computing interaction.

19) Q: DOES THE SYSTEM MAKE ANY PROVISIONS FOR SAVING A PROGRAM AND RE-ENTERING IT AT A LATER TIME?

A: Not mechanically. A user can, of course, as we did here, "Type all" and retype in. We plan very shortly, since the system is still evolving, to put in a "Punch" (on cards) and a "Reload" capability. JOHNNIAC's only output equipment is a 519 punch and an Analex printer; it has a collator on the input end. The system operates now without an operator present.

20) Q: HOW BIG IS THE JOHNNIAC PROGRAM FOR JOSS?

A: Right now we have 6000 40-bit words that are associated with the system. JOHNNIAC, in addition to being small and slow, has no indexing, no floating point (which we wouldn't want anyway), and no indirect addressing. It has the unfortunate characteristic that there is a left instruction and a right instruction in each word, à la 701.

However, the addressing scheme doesn't know about this, so the program is continually plagued with left-right problems. I don't know how much this would condense down to if you had a nice, powerful instruction set, but it could be quite a bit less.

21) Q: DOES THE SYSTEM EMPLOY LINE BUFFERS FOR EACH STATION?

A: Yes. Each station is tied to the buffering system over an open-wire telephone line which is strung by the telephone company. This must be a full-duplex line, because we're sending signals both upstream and downstream at the same time. For instance, if JOSS is typing and the user interrupts, the station must send a signal back indicating that the individual console user wants control. The buffering system consists of a small drum which contains a number of control words--one for each station. In addition to these control words, there are sixteen line buffers which will take 81 characters each. A line buffer assembles a line as it is typed by the user, and takes care of the backspacing, strikeovers, and corrections. When the carriage return is pushed, this sets a bit in the control word in the buffering system which is then examined by the JOSS supervisory program. There's no interrupt system in the JOHNNIAC, but software control of I/O function is preferable because of the interpretive nature of the system. At each interpretation cycle, the interpreter system knows to go around and look at the buffering system. The 16 buffers are assignable to any stations via the control words, which specify which buffer is to be assigned to which station. If everybody is typing in, of course, you need only one buffer line per user. On the other hand, you may have, say, five stations on-line

and operating. Three of them may be in the conversational mode. That means there are three out of the 16 buffers used for input lines. The other 13 may be assigned to the two remaining programs for output lines. This leads to the fact that after we push the INTERRUPT button and JOSS is typing out, we may get four or five lines. This varies considerably. Sometimes you'll interrupt right away, sometimes you'll have filled up a number of buffers before you interrupt.

22) Q: WHERE DID YOU GET THE BUFFER SYSTEM?

A: We made it ourselves. All the hardware for this system, with the exception of the typewriter itself, was constructed at RAND over a period of ten years, the JOHNNIAC early (1952-53) and the consoles late (1961-62).

23) Q: WHAT ARE THE BUFFERS MADE OF?

A: We use a very small drum, made for us to our specifications. All of the electronics was done at RAND, and is reasonably modern transistor circuitry.

24) Q: WHAT ADDITIONS OR CHANGES WOULD YOU LIKE TO MAKE TO THE SYSTEM?

A: We are considering additions to both the language and the hardware. For instance, we'd like to have the "sum" function: sum, over an index running between two limits, the values of an expression. We plan to add a number of functions like this if we can squeeze them in. Field definitions in forms should be made more flexible, too. We would like the ability to save a program in some way: on paper tape, disk file, or in cards for reloading at a later time.

With regard to the console itself, on the Model 868 typewriter we're using, the up/down shift doesn't lock when

JOSS takes control, so sometimes it is possible to shift before control has been returned, and the shift character is lost. This we'd like to fix, and we're investigating the use of the Selectric typewriter. The size of the console electronics could certainly be reduced.

25) Q: HAVE YOU THOUGHT OF IMPLEMENTING THE SYSTEM SO THAT UNUSED COMPUTER TIME COULD BE USED FOR PRODUCTION?

A: Lots of organizations are working on that. We decided to keep our problems simple.

26) Q: DO YOU HAVE ANY IDEA AS TO WHAT THE CAPABILITIES OF JOSS WOULD BE ON A FASTER, LARGER MACHINE?

A: Yes. Just before our 7090 left to be replaced by a 7040/44 combination, a routine was added to the 7090 FORTRAN monitor which would print out statistics, etc., on the nature of jobs, amount of core storage used, time of execution, time of compilation, etc., for each open-shop FORTRAN job that was run through our batching system. We looked over these statistics, and someone made the following observation: "If we had the same processor behind JOSS that we do behind our FORTRAN open-shop system, it looks as if over 50 per cent by number of our FORTRAN open-shop jobs would be handled better in the JOSS interpretive mode than in the FORTRAN mode." And this is just on a straight-time and execution basis. It is interesting to know that JOSS could handle about half of your jobs by number (which is certainly much less by time). I feel that this is a misleading figure, however, for the following reasons: I think that we came up with an average time in a 7090 FORTRAN job of, say, 40 seconds for compilation and perhaps 10 seconds for execution. We feel--we have no way of measuring, of

course--that of this 40 seconds of compilation, perhaps 35 seconds are spent compiling statements which were compiled on a previous run, and 5 seconds are spent compiling out the changes to correct a bug. Similarly on execution: of that 10 seconds, 8 seconds may get you back to where you quit in the previous run, and 2 seconds are spent going from the last bug to the next bug. We don't know how to measure this, but we feel that if we could, this would put the figure even more in favor of the JOSS system.

27) Q: HOW MUCH WOULD USERS PAY FOR THIS SERVICE?

A: JOHNNIAC is free, but I think that they'd pay a lot. We really don't know. This is one of the things we want to determine. Incidentally, the consoles that you see here are rather expensive. The purchase price of those, and the associated electronics, runs around $4000 in a lot size of eight--much more expensive than a desk calculator. And on to that, of course, you have to tack the $75 a month typewriter rental. At any rate, we know that the users are willing to pay enough so that we are going to be forced to provide a follow-on to the JOSS system very shortly. We've done some preliminary work on this sort of thing. We estimate that the follow-on system should have the following characteristics: Roughly 2000 JOSS words available to the user, which would be, at 10 characters per word, around 20,000 characters. At present, it's 1000 characters, so this is a factor of about 20 in core size. Speed should be increased proportionately. I think that we would plan for roughly a hundred terminals to be connected, and service up to something like 25-30 at a time. In addition to this, we would provide long-term storage of some nature. The purchase

price for the new model could, I think, he brought down to somewhere between $500,000 and $750,000. Now this means that the cost per terminal is around $7500, which is high, but perhaps you can put background work on it during the "off" hours. We haven't thought about putting it on during the "on" hours because we feel that response is very important and we've got another computer in the other room anyway that works awfully well. That's about as far as we've gone into the economics of it, but I think that would be an economical situation for almost any engineering or scientific organization with this sort of problem.

JOSS WORDS AND SYMBOLS

COMMANDS

Direct only:
- Cancel
- Delete
- Go
- Form

Indirect only:
- To
- Done
- Stop
- Demand

Both:
- Type
- Do
- Set
- Line
- Page

MODIFIERS

if, and, or
for
in form

DESIGNATORS

step, all steps
part, all parts
form, all forms
all values
all
size

FUNCTIONS

sqrt
log
exp
sin
cos
arg
max
min
sgn
ip
fp
dp
xp

OPERATIONS

\+
\-
·
/
*
||
()
[]

RELATIONS

=
≠
>
≥
<
≤

PUNCTUATION AND SPECIAL CHARACTERS

. , ; : ' " _ # $?