During the last few years a new phrase "man-machine interaction" has become fashionable in computer literature. The phrase is new, but the concept is not. When, in the early fifties, the operator was manipulating his computer, he was actually continually interacting with the machine via the keys and the lights on the console. With batch-processing techniques, the user is now normally far from the computer. Several methods have been implemented or are under development to bring the man back into close contact with his computer. In particular, the last few years have seen the development of several on-line (also called time-sharing) computing systems.

The availability of graphical interactive devices has opened up a new dimension in the field of the man-machine interaction. The displays allow the user to communicate with the computer in graphic terms and with a very high data display speed. They are the most flexible and fast man-machine interactive device available today.

With general-purpose, fully interactive systems, the computer user is able to define, manipulate, and execute his own algorithms on-line in continuous conversation with the computer. In the field of applied mathematics this kind of interaction is of great interest, in particular for research problems when the user wants to experiment with different formulations and different methods of numerical analysis, and when the feedback of results determines the algorithms, i.e. problems where not only is the answer unknown but also the question is difficult to formulate. Many systems of this kind have been implemented in the past, and at CERN the result of several years of practical work in the field of interactive, array-oriented computing with graphical display is SIGMA.

SIGMA (System for Interactive Graphical Mathematical Applications) is a programming language for scientific computing whose major characteristics are the following:

i) The basic data units are scalars, one-dimensional arrays, and multi-dimensional rectangular arrays; SIGMA provides automatic handling of these arrays.

ii) The calculational operators of SIGMA closely resemble the operations of numerical mathematics; procedural operators are often analogous to those of FORTRAN.
The system is designed to be used in interactive mode on terminals connected to a central computer; it provides powerful facilities for graphical display of arrays in the form of (sets of) curves.

The user can construct his own programs within the system and has also access to a program library; he can store and retrieve his data and programs; he obtains, on request, hard copy in alphanumeric or graphical form.

The SIGMA implementation is laid out as a multi-access time-sharing system using the central processor only for actual computation; the implementation also provides for batch-processing use of SIGMA, the user-written code being the same for both interactive and batch modes.

The rest of this note shows the capabilities of the language through examples.

A simple example

We would like to solve the transcendental equation

\[ x = \cos(x) \]

by successive approximations:

1. \( v = 1 \)
2. \( \text{DO 100 } j = 1,50 \)
3. \( 100 \quad v = \cos(v) \)
4. \( \text{PRINT } v, \cos(v) \)

\[ X = 0.73908513392166, 0.7390851339225 \]

Average

We have an array \( A \) which contains 34 items, representing the number of jobs over 34 weeks at the CERN Remote Input/Output Stations (see also Technology Note D29). We type

\[ B = \text{ARRAY(34,1#34)} \]

to place the integers 1 to 34 in \( B \) to represent time, then

\[ \text{DISPLAY A#B,MAX(SUM(A)))/34} \]

plots these quantities and their averages against time, producing Fig. 1.

Mathematical functions

To compute and display the function

\[ f(x) = 4 + \sin x \times e^{x/2} \quad -\pi \leq x \leq \pi \]

we type

\[ F = \text{ARRAY (100,-PI#PI)} \]
\[ F = 4 \times \sin(X) \times \exp(X/2) \]
\[ \text{DISPLAY F % X} \]

This results in the display shown in Fig. 2. Scale of the coordinate axes and position of the coordinate origin are automatically chosen: the system scans through all values of \( F \) and \( X \) and then sets the scale on the axes such that no point is lost while the curve appears in reasonable size.
Fig. 2 Automatic display of the function

\[ f(x) = 4 + \sin x \cdot \exp(x/2) \] for \(-\pi \leq x \leq \pi\)

We might wish to have the same in logarithmic scale and we might also wish a grid. The commands are:

:LOGY
:GRID
DISPLAY F\%X

Finally we wish to see \( F \) and its square together in one single figure without grid and in linear scale; \( F \) as a full line, \( F^2 \) broken. We type

:NOGRID
:LINY
F2 = F ** 2
DISPLAY F\%X, [.-5] F2

Fig. 3 Automatic display of the function \( f(x) \) (as in Fig. 2) with grid and logarithmic scale as the ordinate

Fig. 4 Automatic display of the function \( f(x) \) (as in Fig. 2) together with its square (linear scale, no grid) \( f(x) \) as full line, \([f(x)]^2 \) as broken line.
The following subroutine produces a happy face (Fig. 5) in response to

```
CALL HPYFACE
```

**Fig. 5** A happy face and its program

Recursive functions

Although the procedural aspects of SIGMA are modelled on those of FORTRAN (subroutines, functions, DO-loops), there is one very significant procedural extension: the ability to code recursive routines, i.e. routines which call themselves. Recursively called procedures are terminated in exactly the opposite order to that in which they were begun.

The two examples here, FIB and HANOI, calculate respectively successive terms of the Fibonacci series, where each term (from the third) is the sum of the two preceding terms, and the number of moves needed in the game of the Tower of Hanoi for a given number of disks.

```
1. FIB(I,I=1,20)
2. FIB(I)=FIB(I-1)+FIB(I-2)
3. END
```

This latter game involves transferring a pile of disks of increasing size (top to bottom) from one spike to another, using a third spike as intermediary. Only one disk may be moved in one move, and no larger disk may rest on a smaller one at any time. The legend is that, somewhere in Tibet, a group of monks has been playing one game for thousands of years (with presumably a very large number of disks), and that when they finish the world will end.

```
FIB(1),I=1,20 gives
1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597,
2584,4181,6765
```

and

```
HANOII(I),I=2,4,6,...20 gives
3,15,63,255,1023,4095,16 383,65 535,262 143,1 048 575.
```

A demonstration of SIGMA will be given on a fast graphic display terminal.

**References**

Further information on SIGMA can be obtained from C.E. Vandoni, Tel. 3355; J. Reinfelds, Tel. 3348; or G. Barta, Tel. 5010 (DD Division, CERN).

29 March 1974