CHAPTER I

WHAT IS MATH-MATIC PSEUDO-CODE

MATH-MATIC pseudo-code is a set of words, numbers, and symbols arranged to give a complete, logical statement of a problem. The primary unit of pseudo-code is the sentence. There are three types of sentences; input-output, control, and equation. Input-output sentences cause data to be read into the memory from tape, or information in the memory to be written on tape. Control sentences determine the various paths taken through the program during running. Ordinarily, the sentences, which are numbered in ascending order, are executed in that order. Control sentences can alter this chain of execution in any way the user desires.

Equation sentences are stated as explicit algebraic equations subject to certain conventions listed in Chapter III. The left member of the equation must be the dependent variable, followed by an equal sign. On the right side of the equation appear the various mathematical functions of the independent variables that are needed to calculate the value of the dependent variable. Some examples of equation sentences follow;

(10) X = (15*Y+3*Z)/SIN A .

(6) X(I) = A(I)+B(J,I) .

(11) VARIANCE = SUMSQUARES/10-MEAN^2 .

Input-output and control sentences take the form of English imperative statements. The first word of the input-output or control sentence tells the system the general type of command being given. The remaining words in the input-output or control sentence give further details about this comm-

and, and supply the names of relevant parameters and variables. Some examples of input-output and control sentences follow;

(1) READ A B C .

(20) IF X > Y JUMP TO SENTENCE 8 .

(12) EXECUTE SENTENCE 4 THRU 8 .

(6) VARY J 1 (1) 20 SENTENCE 11 THRU 15 .

Certain conventions and rules regarding the insertion of spaces, parentheses, and periods into input-output and control sentences are necessary for the system to interpret the pseudo-code correctly. These rules are listed in detail in Chapter II. The user should note the flexibility in naming a variable; any single letter or combination of letters and numbers (starting with a letter) up to 12 digits can be handled. Constants may be stated as integers, fractions, decimals, or in power of ten form. One, two or three dimensional arrays of numbers can be read in from tape or constructed in memory. The elements of the array are referred to by the familiar notation of subscripts; these, in turn, may be variables or constants. Four examples follow in which typical problems are described and the Math-Matic pseudo-code statements of the problems are given.

Sample Problem 1:

Solve:

$$Y = \frac{X^3(2+X)}{3\ \mathrm{Cos}\ A} - \sqrt[4]{3F}$$

for F running from 0.2 to 0.8 in increments of 0.2, A running from 0.35 to 1.05 in increments of .175 and X running from 1.8 to 3.3 in increments of 0.5.

A MATHEMATIC pseudo-code statement of this problem is as follows:

(1) VARY P 0.2 (0.2) 0.8 SENTENCES 2 THRU 5 .

(2) VARY A 0.35 (0.175) 1.05 SENTENCES 3 THRU 5 .

(3) VARY X 1.8 (0.5) 3.8 SENTENCES 4 THRU 5 .

(4) Y = X**3*(2+X)/(3*COS A)+4 ROOT (3*P) .

(5) WRITE AND EDIT Y X A P .

(6) STOP .

Sentence 1 will set P to its initial value, 0.2, and will insert following sentence 5 a control operation which will add the increment, 0.2, to P and return control to sentence 2. When P exceeds its limit value of 0.8, control will jump to the next operation following this control, in this case sentence 6. Sentences 2 and 3 will perform similar functions for A and X. The range components in these three sentences indicate that sentence 3 lies within the range of sentence 2, and sentence 2 lies within the range of sentence 1. This nesting of loops means that X, the variable of the innermost loop, will take on all of its values, for each value of A and P. When the value of X exceeds its limit value, A will be incremented and X will be reset to its initial value. P will be incremented and A and X will be reset, each time A exceeds its limit. In this way 100 values of Y will be computed and written. Sentence 6 will supply the necessary sentinels for the output and stop the program.

Sample Problem 2:

We have m samples of 10 values each of a statistical variate. A sentinel of 2's follows the last sample on the tape. We wish to calculate the mean and variance of each sample and edit the output for a uniprinter.

$$\text{MEAN} = \frac{\sum_{i=1}^{10} X_i}{10}$$

$$\text{VARIANCE} = \frac{\sum_{i=1}^{10} X_i^2}{10} - (\text{MEAN})^2$$

We will let the system allocate the input and output servos. The Math-Matic pseudo-code for this problem would be:

(1) READ ITEM X(10) IF SENTINEL JUMP TO SENTENCE 11 .

(2) SUM = 0 .

(3) SUMSQUARES = 0 .

(4) VARY I 1 (1) 10 SENTENCE 5 THRU 6 .

(5) SUM = SUM+X(I) .

(6) SUMSQUARES = SUMSQUARES+X(I)^2 .

(7) MEAN = SUM/10 .

(8) VARIANCE = SUMSQUARES/10-MEAN^2 .

(9) WRITE AND EDIT FOR UNIPRINTER MEAN VARIANCE .

(10) JUMP TO SENTENCE 2 .

(11) STOP .

Sentences 1 and 9 are input-output statements; sentences 4, 10, and 11 are control statements. The rest are all equations. The 10-quantity array read in by sentence 1 will occupy 20 words of storage, since all numbers inside the Math-Matic system are in 2 word floating decimal form. Therefore, every time sentence 1 is executed, the next 20 word item is moved to the current item position. These items are automatically read from the input tape in 60 word blocks, whenever necessary. The user should consider the input items available one at a time as the READ sentence is executed

The VARY in sentence 4 gives I an initial value of 1 and inserts after the sentence 6 a routine which will increment I by 1 and return control to sentence 5, until I exceeds the limit, 10; then control passes to sentence 7. That is, sentences 5 and 6 are executed 10 times, with I = 1, 2, ..., 10. The system automatically places a sentinel on the output tape following the last valid output item, checks the readability of the output tape, and prints out the number of blocks of output on the tape. The superscript symbol 2 which appears in sentences 6 and 8 can be unityped. The method of unityping them is discussed later in the chapter.

Sample Problem 3:

A is a 50X12 matrix, and B is a 12X20 matrix. We want to calculate matrix C = AB. We wish to produce two sets of output, one edited for printing and the other unedited for other calculations. We also wish to print out the element of B with the largest absolute value. We prepare the 240 elements of B on a tape arranged continuously, one row after another. The individual rows of A follow on the same tape and a sentinel of Z's is placed in the first word after the last valid item. We have arbitrarily chosen Servo 3 for the input tape, and we will allow the system to select the output servos for us. The following is a Math-Matic pseudo-code for this problem:

(1) READ ARRAY B(12,20) SERVO 3 .

(2) LARGEST = 0 .

(3) READ ITEM A(12) SERVO 3 IF SENTINEL JUMP TO SENTENCE 14 .

(4) VARY I 1 (1) 20 SENTENCE 5 THRU 8 .

(5) L(I) = 0 .

(6) VARY J 1 (1) 12 SENTENCE 7 .

(7) D(I) = D(I)+A(I,J)*B(J,1) .

(8) IF |D(I)| > LARGEST, JUMP TO SENTENCE 12 .

(9) IGNORE .

(10) WRITE-ARRAY D(20) .

(11) WRITE-ARRAY CONVERTED D(20) .

(12) JUMP TO SENTENCE 3 .

(13) LARGEST = D(I) .

(14) JUMP TO SENTENCE 9 .

(15) PRINT-OUT LARGEST .

(16) STOP .

Sentences 1, 3, 10, and 11 are input-output sentences; 2, 5, 7, and 13 are equations, and the rest are control sentences. Sentence 1 reads in the entire B matrix; sentence 3 reads in one row at a time of the A matrix. When the sentinel item of A is reached the code shifts to finishing up operations. Sentence 2 gives an initial value to the current largest element. Sentences 8, 12, and 14 keep "LARGEST" up-to-date by substituting for it any element whose absolute value is greater. Sentence 5 gives each "D" element a starting value of zero before the summation in sentence 7 calculates the true value. Sentence 4 gives I an initial value of 1, and places after sentence 8 a routine which will increment I by 1 and return to sentence 5 until I exceeds the limit, 20; then this routine lets control pass to the next sentence. That is, sentences 5 to 8 are repeated for I = 1, 2, ..., 20. Similarly, sentence 7 is repeated for J = 1, before

control moves on to sentence 8. Sentence 15 prints out, in floating decimal form, the largest element of D and sentence 16 tells the system the problem is finished. The system then automatically checks the readability of the information on the output tapes and prints out the number of blocks on each. The word "CONVERTED" in sentence 10 means edited with a properly placed decimal point. The system assumes WRITE's (WRITE-ITEM's WRITE-ARRAY's) are for high speed printer unless otherwise specified.

Sample Problem 4:

We have 2 input tapes; one contains n sets of values of A, B and C with a sentinel of Z's after the last set. Beginning with the block after the sentinel, this same tape contains m sets of values of F, G, H and N. Again a sentinel follows the last set. The other input tape contains p sets of values of D and E; p = max(m,n). We wish to evaluate X_1 where,

$$X_1 = \frac{(7000 * Y * A * \sin \alpha)^2}{B^D + C^E}$$

where the value of α is typed in during the problem run. Y assumes all the values 1.0, 1.1, 1.2, ... 3.0 for each set of values of A, B, C, D, and E, and the first set of values of D and E goes with the first set of A, B, and C.

We also wish to evaluate X_2 where,

$$X_2 = \frac{\sqrt[3]{E \cdot G} + \log 10 (D!N)}{F^{2.6} \cdot H}$$

where the first set of values of D and E now goes with the first set of values of F, G, H, and N. Edited output is desired, consisting of A, Y, D, E and X_1 in the first case, and F, D, E, and X_2 in the second. Only one output tape is to be used.

The following is a Math-Matic pseudo-code for this problem;

(1) TYPE-IN ALPHA .

(2) READ A,B,C SERVO 4 STORAGE A IF SENTINEL JUMP TO SENTENCE 8 .

(3) READ D E SERVO 5 .

(4) VARY Y 1 (0.1) 3 SENTENCE 5 THRU 6 .

(5) X1 = (7*10^{3}*Y*A*SIN ALPHA)^{3}/(B PLUS D/C PLUS E) .

(6) WRITE AND EDIT A,Y,D,E X1 SERVO 6 .

(7) JUMP TO SENTENCE 2 .

(8) CLOSE-INPUT AND REWIND SENTENCE 3 .

(9) CLOSE-OUTPUT SENTENCE 6 .

(10) READ F G H N SERVO 4 STORAGE A IF SENTINEL JUMP TO SENTENCE 15 .

(11) EXECUTE SENTENCE 3 .

(12) X2 = (3 ROOT (E-G)+LOG (D+N))/(F$^{2.6}$*EXP H) .

(13) WRITE EDIT F D E X2 SERVO 6 .

(14) JUMP TO SENTENCE 10 .

(15) STOP .

Sentences 1, 4, 7, 14 and 15 are control statements. Sentence 5 and 12 are equations; the rest are input-output statements. Sentence 8 will rewind servo 5 and reset Sentence 3 so that the next time it is executed the first D, E pair will be brought into storage. Sentence 9 clears whatever output items remain in memory on to servo 6 and places a sentinel on that tape, so that another output may be written on it. Sentence 11 causes sentence 3 to be executed, after which control goes to sentence 12. It would have been permissible, and more efficient to repeat the READ instruction of sentence 3 at sentence 11. READ's in this pseudo-code are ordinary READ's, not READ-ITEM's or READ-ARRAY's, because the variables are listed singly rather

than being grouped in subscripted arrays.

A number in power of ten form appears at the beginning of sentence 5 and a decimal exponent appears at the end of sentence 12. The unityping of these superscript exponents is discussed in the next section of this chapter. The specific rules for the expressions in control, input-output, and equation sentences appear in the next chapter.

The letters, digits and symbols appearing in the sentences of the sample problems are all on the ordinary unityper keyboard in the same form except for the superscripts and the greater than (>) and less than (<) symbols. These 15 symbols (including superscript minus, decimal point and slash) have equivalents on the ordinary keyboard whose pulse patterns will be correctly interpreted by the system. For example, superscript 2 is represented by a "%" on the keyboard. A list of these equivalents appears in the appendix to this manual. However, a modified keyboard may be obtained from Remington Rand, Univac which will produce the intended superscript or other special symbol on the typed copy. The use of the list of equivalent for >, <, and numerical exponents brings every MATH-MATIC pseudo-code within range of the ordinary Unityper keyboard. No special training is required to type pseudo-code or to check it.

The experienced programmer will occasionally desire to write a program for a special problem which goes beyond the present MATH-MATIC repertoire. There are two types of code other than MATH-MATIC pseudo-code which may be employed; the intermediate code of MATH-MATIC, known as Arith-Matic pseudo-code, and the familiar Univac code, called C-10. A group of lines of

Arith-Matic pseudo-code may be inserted into the body of a MATH-MATIC pseudo-code by means of a "COMPILER" sentence. A "COMPUTER" sentence causes a similar insertion of any number of lines of Univac C-10 code. These are discussed thoroughly in Chapter V. Both COMPUTER and COMPILER sections may refer to any variable appearing elsewhere in the problem or to any constant by means of a directory. Examples and rules of the two sections and the directory appear in Chapter V. Because of the flexibility and scope of the MATH-MATIC pseudo-code, these special sections will rarely be needed, and the Chapter on them may be safely omitted by those not familiar with UNIVAC.