October 13, 1960

FILE MEMORANDUM

SUBJECT: Proposal for a Comprehensive Matrix Computation System for the STRETCH Computer

I. Objectives

In keeping with the current trend towards more nearly automatic programming, this proposal describes a comprehensive system for carrying out a wide variety of matrix operations as specified by source language statements whose operands are entire matrices rather than individual matrix elements. In view of the expectation that the STRETCH computer will be extensively used for matrix computations, such a system could prove to be a powerful programming aid.

The principal objectives which the proposed system seeks to achieve are as follows:

A. To provide programming facilities for executing such standard matrix operations as:

1. Addition, subtraction, multiplication, and inversion.
2. Transposition, partitioning, and rearrangement.
3. Solution of simultaneous equations.
4. Conversion from one matrix form to another.
5. Computation of eigenvalues and eigenvectors.
6. Input/Output.

B. To permit matrices of several different types to appear as operands in any matrix operation, using a uniform source program notation.

C. To provide optimum computational efficiency and also to conserve storage space by taking full advantage of such structural peculiarities of matrices as numerous zeroes and/or duplicate elements.

D. To provide means for extending the system facilities with respect both to additional matrix types as operands and to additional matrix operations.

This system is intended to work in conjunction with, and eventually to become an integral part of, the STRETCH Macro Language Processor. At the present stage of development of this Processor, it is appropriate to refer only to the Autocoder language as a source language for the proposed matrix computation system. In future, however, means either of extending FORTRAN or of developing a new computing language to supplant FORTRAN may be considered.

In the proposal to follow, which is based on a similar matrix computation system partially developed for the IBM 704 computer, only the main features of the system will be described. Numerous problems of implementation and usage will undoubtedly arise, some of which may have been anticipated, but these problems will not be treated in detail in the proposal.

II. Implementation

A. Autocoder Formats

1. Each matrix will be assigned a symbolic NAME which will always be used without subscripts when referring to the entire matrix as an operand. However, single elements, rows, or columns may be designated by using subscripts in the following manner. For a matrix named ABLE:

the I, J-th element is:	ABLE (I, J)
the I-th row is:	ABLE (I, .)
the J-th row is:	ABLE (. , J)

2. A declarative statement will be used to define each matrix as to type and maximum dimensions. This information will be used by the Processor to make the appropriate entries into the data definition table and to generate the necessary data definition and/or data reservation instructions in STRAP code.

Format:

NAME MATRIX, ELEMENT, STRUCTURE, MAXROW, MAXCOL

Following the matrix NAME, the first field, MATRIX, is the OPCODE and signifies the definition of a matrix. The second field describes the type of matrix element, such as REAL, COMPLEX, etc. The third field describes the structure, such as DIAGonal, SYMmetric, etc. The fourth and fifth fields designate the maximum number of rows and columns of the matrix. For example, a real (single precision floating point), symmetric matrix named DOG, and having maximum dimensions 150 by 150, would be defined thus:

DOG MATRIX, REAL (SP), SYM, 150, 150

(The suggested types of element and structure to be allowed will be discussed in Section III below.)

3. A typical imperative statement will have the format:

OPCODE, NAME 1, NAME 2, NAME 3

when three operands are required. Those arithmetic OPCODES which are appropriate for matrix operations will be retained without change. For example, if MPY is the OPCODE for multiplication, then

MPY, ABLE, BAKER, CHARLIE

will mean that the matrix CHARLIE is the product of ABLE * BAKER. In the case of matrix inversion, however, only two operands are required. Hence,

INVERT, DOG, EASY

will mean that the matrix EASY is to be computed as the inverse of DOG. (The suggested OPCODES will be described in Section IV below.)

4. Each imperative statement of the foregoing type will be translated by the Processor into a STRAP calling sequence which will be used to gain entry into the appropriate subroutine(s) to execute the operation requested. This calling sequence will be equivalent in information content to the Autocoder instruction.

B. Memory Block Structure

1. All numerical data for each matrix will be stored consecutively beginning with the third word of a memory block assigned to that matrix.

2. The first two "header" words will be reserved for descriptive information. The first header word of each such block, which is at the address symbolized by the matrix NAME, is a control word having its own address in its address field and a count of the total number of words in the block in its count field. This word will be used for input/output operations.

3. The second header word of the block will consist of two half-words with the actual number of rows stored in the first 18 bits of the address field of the first half-word and the actual number of columns similarly stored in the address field of the second half-word. The remaining bits of these two half-words are available for the type code of the matrix.

4. Each different type of matrix will have its own peculiar storage format, chosen so as to conserve storage space both in core memory and on tape. As a consequence, a number of different subroutines will be required to execute each type of matrix operation. But by eliminating trivial computations with zeroes and/or duplicate elements, much greater computational efficiency can thus be achieved.

C. Method of Execution

1. All matrix operations will be executed interpretively using the STRAP calling sequence to gain entry into the appropriate master subroutine for carrying out each different operation such as add, invert, etc.

2. The initial task of each such master subroutine, immediately after entry is gained, will be to examine the second header word of each of the operand matrices in order to determine the types of matrix involved in the operation to be executed. By means of a table look up, the correct inner subroutine required to execute the operation is then chosen.

3. Before transferring control to this inner subroutine, however, the master subroutine first checks the dimensions of the operand matrices to ensure that they are compatible with the operation to be performed.

4. When control has been passed to the inner subroutine, the first task is that of forming the two header words for the result matrix. The actual operation is then performed and upon its completion, control is returned to the next calling sequence in the main program.

5. Error messages will be printed in cases where no inner subroutine exists for carrying out the called-for operation on the types of matrix specified or in case the matrices have incompatible dimensions.

III. Matrix Types and Storage Formats

A. Elementary and Special Matrices

1. Certain special matrices, which are useful computational or symbolic tools, will be allowed. The following elementary matrices require no numerical data and can be completely described by the two header words.

   a. Null matrix. (The dimensions will be given in the second header word, if needed. If the dimensions are omitted--i.e., set to zero--this will be taken to imply that the null matrix is of adjustable size.)

   b. Unit matrix. (Again, dimensions will be given if needed--if omitted, the size will be considered adjustable.)

   c. All-ones vector. (Dimensions n x 1 for a column vector and 1 x n for a row. If n = 0, this dimension will be considered adjustable.)

   d. Unit row vector. (First half of second header word gives length of vector; second half word gives position of the only nonzero element, which is unity. If length = 0, it will be considered adjustable.)

   e. Unit column vector. (Similar to above.)

f. E_{ij} matrix--all zeroes except for unity in the i-th row and j-th column. (Dimensions adjustable.)

Additional matrices, related to some of those just described, are the following in which a constant (real or complex, simple or multiple precision floating point) is stored following the two header words.

g. Scalar matrices (k * unit matrix)

h. k * all-ones vector

i. k * unit row or column vector

j. k * E_{ij}

In each of these cases, the previously described conventions regarding fixed or adjustable dimensions will apply.

2. Other special matrices of an elementary type are the following:

a. Binary matrices--consisting of 1 or 0 elements only. (Stored by rows or by columns, 64 bits per word, each row or column starting at a full word boundary and occupying as many consecutive words as needed.)

b. Ternary matrices--consisting of +1, -1, or 0 elements only. (Stored by rows or by columns, 64 bits per word, each row or column starting at a full word boundary. The magnitudes of the first 64 elements of each row or column are stored in the first word and the corresponding signs in the second word of the row or column--followed by as many alternating words of magnitude bits and sign bits as needed.)

c. Incidence matrices--tabular form. (These matrices are used to describe connectivity or incidence relations between branches and nodes in a linear graph. Each data word, corresponding to a directed branch of the graph, will contain the number of the initial node in the address of the first half word and that of the final node in the address of the second half word. This form may be converted to a ternary matrix, if desired.)

B. Matrix Classification according to Type of Element

The elements of matrices may be not only null, binary, or ternary numbers as described above, but also any of the following data objects:

1. Real numbers, single or multiple precision floating point. (Stored as normalized binary floating point, if single precision. If multiple precision, the most significant portion is stored in the first word of a block in normalized form, followed consecutively by unnormalized words having successively lower significance.)

2. Complex numbers, single or multiple precision floating point. (Stored consecutively with real part immediately preceding imaginary part. If multiple precision, both the real and imaginary parts are stored in two consecutive blocks each having the same format as that of a multiple precision real number.)

3. Polar numbers (polar form of complex numbers), single or multiple precision floating point. (Stored consecutively with modulus part preceding argument or phase.)

4. Submatrices, designated by the symbolic NAMEs associated therewith. (Stored with one submatrix NAME per word in A8 code. Matrices whose elements are other matrices are known as compound matrices. With this classification included in the system, compounding of matrices to any depth is possible, at least in principle, and can be used to handle multisubscripted variables.)

C. Matrix Classification according to Structure

The following structural classification, subject to later extension, includes the most common types encountered in scientific and engineering applications of matrices.

1. Rectangular matrix. (Stored consecutively by rows or by columns. Includes row and column vectors as special cases.)

2. Symmetric matrix (real) or Hermitian (complex). (Stored consecutively by columns omitting all elements above the main diagonal.)

3. Diagonal matrix. (Only the diagonal elements are stored.)

4. Codiagonal matrix. (Stored consecutively as follows: main diagonal, first subdiagonal, first superdiagonal, second subdiagonal, second superdiagonal, and so on until all nonzero codiagonals have been stored. If symmetric, only subdiagonals are stored. Type code must indicate number of codiagonals. If this number is zero, all codiagonals will be stored.)

5. Hessenberg matrix. (Contains one superdiagonal, main diagonal, and all subdiagonals. Stored with superdiagonal first, main diagonal next, followed by consecutive subdiagonals.)

6. Upper or lower triangular matrix. (Stored with main diagonal first followed by successive sub- or superdiagonals.)

7. Sparse matrices. (Only the nonzero elements will be stored, in sequences similar to those of the rectangular, symmetric, or codiagonal matrices described above. In addition, a key showing the location of these nonzero elements in the matrix would be needed. Two alternative methods for storing this information are:

   a. A bit pattern wherein 1's indicate the present of nonzero matrix elements.

   b. Two half words, following each nonzero data word, and showing the row and column position thereof.)

IV. Recommended Autocoder Instructions for Matrix Computation

A. Addition, Subtraction, and Multiplication

In keeping with the previously suggested Autocoder formats (see File Memo, September 7, 1960, by F. H. Branin), each matrix operand may be prefixed by a minus sign, if desired. (The slash, representing absolute value, will not be used.) Accordingly, if we use the more conventional opcodes ADD, SUB, and MPY, we may express the operations of addition, subtraction, and multiplication of matrices as follows:

ADD, A, B, C	means	A + B = C
ADD, A, -B, C	means	A - B = C
SUB, A, B, C	means	A - B = C
MPY, -A, B, C	means	-A * B = C

Arithmetic operations on single rows, single columns, or single elements of matrices may be specified using subscripts as follows:

ADD, A(I, .), B(K, .), C	where C is a row vector
MPY, A(I, .), B(. , L), C	where C is a scalar
MPY, A(. , J), B(K, .), C	where C is a matrix
SUB, A(I, J), 5. , C	where C is a scalar

B. Inversion

INVERT, A, B	means	$A^{-1} = B$
INVERT, -A, B	means	$-A^{-1} = B$

C. Transposition

XPOS, A, B	means	$A_t = B$

D. Partitioning and Rearrangement

In partitioning a given matrix by extracting a contiguous sequence of rows and columns without rearranging them, it is sufficient to specify the first and last rows and the first and last columns. Thus, if we desire to extract the 5-th through 11 th rows and the 2nd through 4-th column of matrix A and designate this new matrix as B, we may write the three Autocoder instructions:

PARTIT, A, B
ROWS, 5-11
COLS, 2-4

If we wish to extract the K-th through L-th rows and M-th through N-th columns, we may write:

PARTIT, A, B
ROWS, K-L
COLS, M-N

where K L and M N, of necessity.

If a rearrangement is desired, either with or without partitioning, we may use a similar format but with the rows and columns of the matrix to be partitioned listed explicitly in the order in which they are to appear in the result matrix. For example, if we wish to let rows 5, 2, 7, and 9 of A be the first through fourth rows of B and

columns 2, 4, 1, 7, and 11 of A be the first through fifth columns of B, we may write:

PARTIT, A, B
ROWS, 5, 2, 7, 9
COLS, 2, 4, 1, 7, 11

In either usage, all three instructions must always appear together with PARTIT first. If the sequence is PARTIT, ROWS, and COLS, then the rows of the partitioned matrix become the rows of the result matrix, and likewise with the columns. However, if the sequence is PARTIT, COLS, and ROWS, then the columns of the partitioned matrix become the rows of the result and vice versa. In other words, a transpostion is included along with the partitioning and/or rearrangement.

E. Solution of Simultaneous Equations

If the matrix equation AX = B is to be solved, whether X and B are single column vectors, or a matrix consisting of several column vectors, the Autocoder instruction calling for solution of this equation will be

SOLVE, A, X, B meaning solve AX = B

Since matrix inversion requires roughly a factor of three times as much computation as the solution of simultaneous equations, solution by means of matrix inversion will actually be employed whenever X and B consist of four or more columns.

F. Conversion from one Matrix Form to Another

If matrix A is diagonal or symmetric and is to be converted to rectangular form, stored rowwise, which is the form of matrix B, the Autocoder instruction for effecting this conversion is

SET =, A, B

This instruction can also be used to move a matrix from one location to another without change of form if both operand matrices have the same form. If an inadmissible conervsion is called for by mistake--such as converting a rectangular matrix to diagonal form--the operation will be carried out nevertheless, but a diagnostic message will be printed out at the same time.

G. Computation of Eigenvalues and Eigenvectors

No recommendation concerning the Autocoder instruction format for this computation will be made at present. The problem needs to be studied extensively. Suitable instruction formats can be formulated and added to the system later on.

H. Input/Output

Input-output operations with matrices will be called for using the format suggested by R. L. Harding. (See File Memo, October 5, 1960.)

V. Some Recognized Problems and their Possible Remedies

A. The biggest drawback of the proposed system is the large number of inner subroutines that would be required to handle the various different situations that can arise with the matrix types suggested. For example, in the case of real, single precision matrices, alone, if we allow six structural classes (rowwise or columnwise rectangular, symmetric, diagonal, codiagonal, and symmetric codiagonal), 36 different subroutines would be required for multiplication and almost that number for addition or subtraction. Accordingly, some limitation on the allowable combinations of matrix types must be imposed, at least at first while the system is being developed and checked out. At a later date, a scheme could be worked out for scanning all matrix types and operations during compilation of each source program and selecting only those subroutines which are actually needed. In this way, the storage space allotted to the matrix computation system in the object program could be minimized.

B. As new matrix types are added to the initial system, the tables used in selecting subroutines will need to be expanded. Accordingly, a mechanism for effecting this expansion, in conjunction with the STRETCH Macro Language Processor, will have to be developed.

C. In a given operation upon compound matrices--for example, multiplication--the master subroutine which effects the interpretation and inner subroutine selection, will have to be entered repeatedly from within itself--or rather from within one or more of its inner subroutines. Moreover, the inner subroutines themselves may have to be entered repeatedly in similar fashion. Accordingly, an expandable bookkeeping scheme, perhaps in the form of a push-down list, will be needed to keep track of the various points of entry and exit to these various subroutines.

D. Development of suitable Autocoder instructions for various matrix operations should be straightforward, but the data description statements may be a bit more involved.

E. Although each matrix will be stored as compactly as possible, a great deal of core memory space may be unused simply because data reservations are planned, at present, to be made on the basis of the maximum dimensions of each matrix operand. However, it may later be feasible to develop a dynamic memory allocation scheme, for use at execution time, which stores all matrices consecutively as they are encountered in the program and, when memory is about to overflow, saves only those matrices which will be called for later in the program.

F. H. Branin

F. H. Branin

FHB:jas