On a Non-Numeric Matrix Processing Language MPL-1

TAKETOSHI MISHIMA*

Abstract

A new symbol manipulation language suitable for matrix operations is proposed, which has the following features: (1) to be a simple language aimed only at non-numeric or numerical matrix operations; (2) the notation for writing of the operations is the similar form of standard mathematical ones; (3) to require none of special knowledges or techniques in use; (4) the programs written in this language is readable with a little explanations; and (5) the implementation of MPL-1 interpreter is not difficult.

1. *Introduction*

In numerical processes, the difficulties we used to encounter in the practical performance have been peeling offone after another in accordance with the development of higher speed and larger computer systems with higher languages.

On the other hand, comparing with numerical processes, far more difficulties have still left in symbol manipulations which have these merits: (1) results obtained have generality; (2) the results hold full precisions; (3) effective numerical processings are promised after symbolic operations are finished.

In this paper a language is shown, which has made it possible to manipulate non-numeric or numerical matrix operations in the same fashion of processing simple arithmetic ones in an algorithmic language.

2. *Obstacles in Symbol Manipulations and its Countermeasures*

Explosively growing data make the practical symbol manipulations very difficult. A number of operations, in addition, are needed in matrix processings even in case of numerical ones, and following obstacles are brought about: (1) to be required essentially tremendous memory capacity in processings; (2) the results obtained are unavailable for its formidable lengths and complexities; (3) too much time is expended to be economic. These are too complex for the general way to have efficiency. In these cases severely restricted processings alone achieve satisfactory results.

This paper first appeared in Japanese in Joho Shori (the Journal of the Information Processing Society of Japan), Vol. 12, No. 6 (1971), pp. 326~334.

* Faculty of Engineering, Meiji University

3. *The Construction of MPL-1*

3.1 *Basic Symbols, Identifiers, Strings and $-Brackets*

(1) Basic Symbols

⟨basic symbol⟩::=⟨letter⟩|⟨digit⟩|⟨delimiter⟩|⟨sign⟩|$

MPL-1 is built up from basic symbols which have no fixed meaning.

(2) Letters

⟨letter⟩::=A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|
X|Y|Z|*|/|=|♯|&|@|;|:|,|%|(|)|↑

Letters have no inherent meaning and are used for forming identifiers and strings. Hereafter '*I*' is used for the symbol of imaginary unit for convenience' sake.

(3) Signs

⟨sign⟩::=+|−

Signs have their conventional meaning and are also able to serve for forming identifiers and strings.

(4) Digits

⟨digit⟩::=0|1|2|3|4|5|6|7|8|9

Digits are used to construct numbers, identifiers and strings.

(5) Delimiters

⟨delimiter⟩::=,|.|(|)|b

Any strings is separated by delimiters. 'b' indicates a blank. Hereafter one or more blanks are indicated by ';'.

(6) Identifiers

⟨identifier⟩::=⟨letters except *I*⟩|⟨sign⟩|$|$⟨identifier part⟩|⟨letter⟩⟨identifier part⟩

⟨identifier part⟩::=⟨character⟩|⟨character⟩⟨identifier part⟩

⟨character⟩::=⟨letter⟩|⟨digit⟩|⟨sign⟩

Identifiers serve for identification of variables and procedures but have no individual meaning.

(7) Strings

⟨string⟩::=⟨basic symbol⟩|⟨basic symbol⟩⟨string⟩

(8) $-Brackets

⟨$-bracket⟩::=$$⟨a certain basic symbol *X*⟩⟨string⟩*X*

Any basic symbol is able to be used for a constituent of identifiers within $-brackets.

3.2 *Numbers*

⟨number⟩::=⟨real nuber⟩|⟨complex number⟩

Real and complex numbers have their conventional meaning. The detail of numbers are abbreviated for avoiding monotony.

3.3 *Expressions*

Algorithmic processes are described by expressions.

(1) Variables

⟨variable⟩::=⟨identifier⟩

A variable is used to designate a certain value, but two or more different quantities cannot be given for the same variables simultaneously.

(2) Function Designators

⟨procedure identifier⟩::=⟨identifier⟩

⟨actual parameter⟩::=⟨variable⟩|⟨function designator⟩|(⟨arithmetic expression⟩)

⟨argument⟩::=⟨actual parameter⟩|⟨actual parameter⟩; ⟨argument⟩|empty

⟨function designator⟩::=⟨procedure identifier⟩; (⟨argument⟩)

A function designator yields a value for a set of given actual parameters.

(3) Arithmetic Expressions

⟨term⟩::=⟨number⟩|⟨scalar constant⟩|⟨variable⟩|⟨function designator⟩|(⟨arithmetic expression⟩)

⟨scalar constant⟩::=⟨the identifier defined by scalar definition⟩

⟨arithmetic expression⟩::=⟨term⟩; ⟨operator⟩; ⟨term⟩|⟨term⟩; ⟨operator⟩; ⟨arithmetic expression⟩

⟨operator⟩::=*|+|−

⟨matrix⟩::=((⟨element list⟩)⟨matrix part⟩)

⟨matrix part⟩::=;(⟨element list⟩)|;(⟨element list⟩)⟨matrix part⟩|empty

⟨element list⟩::=⟨element⟩|⟨element⟩; ⟨element list⟩

⟨element⟩::=⟨variable⟩|⟨number⟩|(⟨*E*-arithmetic expression⟩)

A rule for computing a value is indicated by arithmetic expressions which are used within definition statements. Table 1 explains the meaning and priority of operators.

Table 1. The Meaning and Priority of Operators.

Operator	Priority	Meaning
*	1st	Matrix Multiplication or Scalar Multipul
+	2nd	Matrix Addition
−	2nd	Matrix Subtraction

(4) *E*-arithmetic Expressions

E-arithmetic expressions have the same syntax and the similar semantics except that they function to the elements of matrices and *E*-operators are different in meaning from operators. Table 2 shows what the *E*-operators are.

3.4 *Standard Functions*

Standard functions are shown in Table 3.

Table 2. The Meaning and Priority of E-Operators.

E-Operator	Priority	Meaning
*	1st	Multplication
/	1st	Division
+	2nd	Addition
−	2nd	Subtraction

Table 3. Standard Functions.

Name	Function
TRANS	Transposed Matrix
INVM	Inverse Matrix
DETM	Expansion of Determinant
ELM	Get a Matrix Element
SN1	Get $(m-1, n-1)$ of (m, n)*
MALPHA12	Get $(X_{1n}\cdots X_{m-1n})^t$ of (m, n)
MALPHA21	Get $(X_{m1}\cdots X_{mn-1})$ of (m, n)
SMAT	Establish a Matrix
COL	Get a Column
ROW	Get a Row
SPUR	Trace
INVERSVECTOR	Inversevector
MINOR	Minor
SIMUE	Give the Solution of $AX=B$
EIGEN	Property Equation
TELM	List of Diagonal Components
EXCOL	Exchange Two Columns Each Other
EXROW	Exchange Two Rows Each Other
ELMEX	Replace an Element by Someone
EMAT	Give an Elementary Matrix
INPROD	Inner Product
EXPROD	Exterior Product on Three Dimensional Vectors

* (m, n) means the matrix of (m, n)-type.

Table 4. Out-Put Functions.

Name	Function
PRINT	Print the Value of its Argument
FLPRINT	Print the Value of its Argument with Each Top Level List

3.5 *Out-Put Functions*

See Table 4.

3.6 *Function ERASE*

ERASE is useful for recovering the expended memory scope.

3.7 *Defined Functions*

Functions defined by definition statements are called defined functions.

3.8 *Statements*

A statement is the unit of operations within the language.

⟨statement⟩::=⟨assignment statement⟩|⟨procedure statement⟩|
⟨definition statement⟩

(1) Assignment Statements

⟨assignment statement⟩::=⟨variable⟩;:=;⟨expression part⟩

⟨expression part⟩::=⟨function designator⟩|(⟨*E*-expression part⟩)|
⟨matrix⟩

⟨*E*-expression part⟩::=⟨*E*-term⟩;⟨*E*-operator⟩;⟨*E*-arithmetic expression⟩

An assignment statement serves for assigning a computed value to a variable.

(2) Definition Statements

⟨definition statement⟩::=⟨identifier⟩;(⟨variable list⟩);=
;(⟨arithmetic expression⟩)

⟨variable list⟩::=⟨variable⟩|⟨variable⟩;⟨variable list⟩

Definition statements are used for composing any procedure.

(3) Procedure Statements

⟨procedure statement⟩::=⟨procedure identifier⟩;(⟨argument⟩)

⟨procedure identifier⟩::=⟨identifier⟩

⟨argument⟩::=⟨argument part⟩|empty

⟨argument part⟩::=⟨actual parameter⟩|⟨actual parameter⟩;
⟨argument part⟩

⟨actual parameter⟩::=⟨variable⟩|⟨number⟩|⟨matrix⟩|⟨function designator⟩|⟨arithmetic expression⟩

Procedure statements are available for out-put functions and ERASE.

3.9 *Declarations*

A certain property of identifiers are declared by declarations. MPL-1 has following two declarations.

(1) Scalar Declarations

⟨scalar declaration⟩::=SCALAR;(⟨identifier list⟩)

⟨identifier list⟩::=⟨identifier⟩|⟨identifier⟩;⟨identifier list⟩

Identifiers declared by scalar declarations are manipulated within definition statements as constants indicated scalar quantities.

(2) Funname Declarations

⟨funname declaration⟩::=FUNNAME;(⟨identifier list⟩)

A certain indicator is put to the identifier declared by funname declarations and the identifier is interpreted as a function call after that.

3.10 *Programs*

MPL-1 full programs have the following format:

$C;\cdots;C;D;\cdots;D;S\cdots S;$END;

where C, D and S are comments, declarations and statements respectively. C and

D are not essential. Comments has the same significance as that of FORTRAN, and END indicates the end of a program.

4. *MPL-1 interpreter*

MPL-1 programs are processed by MPL-1 interpreter programed in a LISP 1.5 dialect.

4.1 *Interpreter*

MPL-1 interpreter is quite a simple one.

4.2 *ERASE*

The expended memory scope is recovered automatically in MPL-1. ERASE has a function of recovering memory scope, too. ERASE, but, manipulates the untouchable memory scope for MPL-1 interpreter to process automatically.

4.3 *FM926*

An extended HELP association list called FM926 is used for processing a part of assignment statements, the rest of which is manipulated HELP property list.

4.4 *Simplifications*

The simplification problem is essential in symbol manipulations. MPL-1 has several groups of simplification routines which act automatically and release us from the troubles of simplifications.

5. *Examples*

The following worked examples will show a part of MPL-1's abilities.

Problem 1: Write a program which will compute the value of F.

$$F=A(A+(B-A)B),$$

where
$$A=\begin{pmatrix} a11+a12\times a13 & a14 \\ a21 & a11+a12\times a13 \end{pmatrix},$$
$$B=\begin{pmatrix} b11 & b21-b13\div b14 \\ b21-b13\div b14 & b24 \end{pmatrix}.$$

Problem 2: What will be the value of N and S?

$$N=APA^T+BB^TQ,$$
$$S=NH^T(HNH^T+R)^{-1}HN,$$

where $H=(1\ 0)$, $A=\begin{pmatrix} a & b \\ 0 & 0 \end{pmatrix}$, $B=\begin{pmatrix} c \\ 0 \end{pmatrix}$, $P=\begin{pmatrix} p11 & p12 \\ p21 & p22 \end{pmatrix}$,

Q and R are scalar quantities.

EXAMPLE 1 and EXAMPLE 2 show the executed results for problem 1 and problem 2 respectively.

6. *Conclusion*

MPL-1 is enable us to process non-numeric or numerical matrix operations

```
C EXAMPLE 1
  F (A B) = (A * (A + (B - A) * B))
  VA := (A11 + A12 * A13)
  VB := (B21 - B13 / B14)
  A  := ((VA A14) (A21 VA))
  B  := ((B11 VB) (VB B24))
  F1 := F (A B)
  PRINT (¥¥¥THE VALUE OF F¥)
THE VALUE OF F
  FLPRINT (F1)
((+ (* (+ A11 (* A12 A13)) (+ A11 (* A12 A13)
  (* B11 (+ B11 (* -1 (+ A11 (* A12 A13)))))
  (* (+ B21 (/ (* -1 B13) B14)) (+ B21 (/ (* -1 B13) B14) (* -1 A14)))))
   (* A14 (+ A21 (* B11 (+ B21 (/ (* -1 B13) B14) (* -1 A21)))
  (* (+ B21 (/ (* -1 B13) B14)) (+ B24 (* -1 (+ A11 (* A12 A13)))))))))
  (+ (* (+ A11 (* A12 A13)) (+ A14 (* (+ B21 (/ (* -1 B13) B14))
  (+ B11 (* -1 (+ A11 (* A12 A13))))) (* B24
  (+ B21 (/ (* -1 B13) B14) (* -1 A14)))))
  (* (+ A11 (* A12 A13) (* (+ B21 (/ (* -1 B13) B14))
  (+ B21 (/ (* -1 B13) B14) (* -1 A21))) (* B24
  (+ B24 (* -1 (+ A11 (* A12 A13)))))) A14)))
((+ (* A21 (+ A11 (* A12 A13) (* B11 (+ B11 (* -1 (+ A11 (* A12 A13)))))
   (* (+ B21 (/ (* -1 B13) B14)) (+ B21 (/ (* -1 B13) B14) (* -1 A14))))
  ) (* (+ A11 (* A12 A13)) (+ A21 (* B11 (+ B21 (/ (* -1 B13) B14)
  (* -1 A21))) (* (+ B21 (/ (* -1 B13) B14))
  (+ B24 (* -1 (+ A11 (* A12 A13)))))))) (+
  (* A21 (+ A14 (* (+ B21 (/ (* -1 B13) B14))
  (+ B11 (* -1 (+ A11 (* A12 A13))))) (* B24
  (+ B21 (/ (* -1 B13) B14) (* -1 A14)))))
  (* (+ A11 (* A12 A13) (* (+ B21 (/ (* -1 B13) B14))
  (+ B21 (/ (* -1 B13) B14) (* -1 A21))) (* B24
  (+ B24 (* -1 (+ A11 (* A12 A13)))))) (+ A11 (* A12 A13)))))
  END
```

EXAMPLE 1 and EXAMPLE 2.

```
C    EXAMPLE 2
  SCALAR (Q R)
  S (N1 H TH) = (N1 - TH * INVM ((H * N1 * TH + R)) * H * N1)
  N (A P B) = (A * P * TRANS (A) + B * TRANS (B) * Q)
  H := ((1 0))
  A := ((A B) (0 0))
  B := ((C) (0))
  P := ((P11 P12) (P21 P22))
  H := TRANS (H)
  N1 := N (A P B)
  S1 := S (N1 H TH)
  PRINT (¥¥¥THE VALUE OF N¥)
THE VALUE OF N
  FLPRINT (N1)
((+ (* A (+ (* A P11) (* B P21))) (* B (+ (* A P12) (* B P22)))
  (* C C Q)) +0)
(+0 +0)
  PRINT (¥¥¥THE VALUE OF S¥)
THE VALUE OF S
  FLPRINT (S1)
((/ (+ (* -1 (+ (* A (+ (* A P11) (* B P21)))
  (* B (+ (* A P12) (* B P22))) (* C C Q)))
  (* (+ R (* A (+ (* A P11) (* B P21))) (* B (+ (* A P12) (* B P22)))
  (* C C Q)) (+ (* A (+ (* A P11) (* B P21)))
  (* B (+ (* A P12) (* B P22))) (* C C Q))))
  (+ R (* A (+ (* A P11) (* B P21))) (* B (+ (* A P12) (* B P22)))
  (* C C Q))) +0)
(+0 +0)
  END
```

in the same manner of processing simple algebraic ones in an algorithmic language. The language implemented for a certain problem only proves fruitful in symbol manipulations for the troubles with it come from the basis of computer systems, then the language which has generality cannot have practical meaning. In view of the current aspects it is hoped that the language suitable for a certain problem is provided soon when it is wanted.

Acknowledgements

The auther is indebted to Professor Motinori Goto, Meiji University, for his helpful suggestions and encouragements. He is also grateful to the late Mr. Akira Kamoi and his followers in Hitachi Central Reserch Laboratory.

References

[1] McCarthy, J. et al.: "LISP 1.5 Programmers' Manual", *MIT Press* (1962).
[2] "HELP Manual", Hitachi Ltd. (1968).
[3] Dean Wooldrige Jr.: "An Algebraic Simplify Program in LISP", Stanford Artificial Intelligence Project Memo No. 11 (1963).