N E W S L E T T E R # 6

March 1982

Aloha from Hawaii! The Soft Warehouse Newsletter provides you with information on new Soft Warehouse products, and software extensions or corrections to existing products. In addition, the newsletter is a medium for the exchange of ideas and application programs within the growing community of **muMATH** and **muLISP** users.

If you would like to subscribe, or extend your subscription to the Newsletter for three issues beyond the expiration number on your mailing label, please send $6 ($10 for orders from outside the U.S. or Canada) by check, VISA, or Master Card to The Soft Warehouse, P.O. Box 11174, Honolulu, Hawaii, 96828, U.S.A. A complete set of back issues is available on request for $15.

The 7th West Coast Computer Faire

This year's West Coast Computer Faire will be held March 19-21 in San Francisco's Civic Auditorium. The Soft Warehouse will be exhibiting muMATH and muLISP in Brooks Hall at booth 1713. We look forward to meeting users of our software who would like to stop by.

muMATH and muLISP for the APPLE II Computer

Now available direct from The Soft Warehouse are full implementations of **muMATH-80** and **muLISP-80** for the "native" Apple II Computer, not requiring Microsoft's Z80[tm] SoftCard[tm]. Included with the software is **ADIOS** (Apple Disk Interface and Operating System). This flexible operating system provides a friendly operating environment for muMATH and muLISP.

Computer Algebra in Scientific American

Computer symbolic mathematics may finally be getting the recognition it deserves. The December 1981 issue of Scientific American contains an excellent article about the subject entitled "Computer Algebra" by **Richard Pavelle**, **Michael Rothstein**, and **John Fitch**. The article discusses the history, motivation, and major areas of study in the field of computer algebra. Naturally most of the article is in reference to the large "mainframe" systems; however, **muMATH** is mentioned as "the most sophisticated and widely available system" for microcomputers. Mathematics and Computer Education (formerly The MATYC Journal) also published a brief review of muMATH in the Winter 1982 edition, Volume 16, No. 1.

The SOFT WAREHOUSE SOFTWARE CONTEST!

muMATH and muLISP are currently being used by a relatively small number of loyal enthusiasts. For the most part, you are experienced programmers who are challenged at the prospect of writing programs for these systems. There is, however, a much larger group of people, not interested in programming, who could benefit from the AI and other application software you produce. Until a computer language has established a sizable inventory of application programs, both public domain and proprietary, it will be only a passing novelty. We do not intend to let that happen to our products.

Therefore, the Soft Warehouse is sponsoring an application software contest. This will publicize existing software and encourage the creation of new software systems written in either muLISP^{tm} or muSIMP^{tm}. This publicity should in turn financially reward and/or enhance the reputation of the authors.

The muSIMP and muLISP entries will be judged in separate categories. A $100 first prize and a $50 second prize will be awarded in each category. To enter, send the following items to reach us by June 1, 1982:

1. A fact sheet containing:
    a. your name, address, and phone number (indicate if you do not want your name and address published in the Newsletter);
    b. the software license number, version number and/or date of your copy of muMATH or muLISP;
    c. the name of your application program;
    d. the diskette format on which the program is provided (e.g. 8" CP/M, Apple ADIOS, Apple SoftCard, TRS-80 Model 1, etc.);
    e. a 1/2 to 1 page description of the program that is suitable for publication in the Newsletter;
    f. a statement declaring the software to be proprietary or in the public domain. If it is proprietary, indicate price (proprietary entrants must be willing to have their name and address published).

2. Diskette(s) containing the software.

3. Complete documentation that includes what the program does, who needs it and why, hardware/software prerequisites, examples of how to use it, a table of contents and index as applicable.

The next SWH Newsletter will announce the contest winners and publish descriptions for the best entries. The entries will be judged on usefulness, ease of use, and documentation.

All rights to the software and documentation remain with the authors. We only wish to serve as a communication link between software authors and consumers. Unless you prefer anonymity, people interested in your program will contact you directly. Otherwise, the Soft Warehouse will be happy to forward information requests, provided your software is in the public domain.

Computation of Determinants using Minor Expansion

The following question came from **Dr. Frederick H. Raab** of Burlington, Vermont: "My problem involves determining the constants K1, K2, and K3 in the expression

DET (L*I - P) = L^3 + K1*L^2 + K2*L + K3

where **I** is an identity matrix and **P** is a symmetric matrix; **L** is a scalar. (You will recognize this as a characteristic-value problem)."

"What needs to be done appears to be a simple set of multiplications, followed by collection of terms according to powers of L. What muMATH gives is a 10-line expression containing all sorts of divisions by factors such as (L-P23). How can muMATH be forced to give a useful answer?"

muMATH's determinant routine uses a variant of 'Gaussian elimination' that entails divisions. muMATH-80 cannot cancel nonnumeric common divisors that are not explicitly represented in a result, and the FCTR function is capable of extracting only simple 'monomial' factors. Consequently, if EXPD (FCTR (EXPD (DET(L*I-P))) does not yield a polynomial form in L, there are nontrivial implicit common divisors that muMATH cannot discover to cancel.

Expansion of determinants by minors does not require division, thus this problem is avoided. A muSIMP source listing for a determinant function using minor expansion is given at the end of this newsletter. For large numeric and/or 'sparse' matrices, this simple variant of minor expansion is dramatically slower than Gaussian elimination. However, for small problems involving nonnumeric elements, you may generally prefer minor expansion -- at least until we figure out a way to make muMATH cancel all nonnumeric common divisors.

Most likely, your resulting polynomial form in L will not have terms grouped exactly as you wish. However, if the form is assigned to a variable D, then

K3: EVSUB (D, L, 0);

after which

K2: EVSUB ((D-K3)/L, L, 0);

or

K2: EVSUB (DIF (D, L), L, 0);

etc. This technique of using substitution and cancellation or differentiation to extract coefficients is useful to know.

Linear Regression Analysis using muMATH

As part of a numerical methods course at the University of Hawaii, **Stuart Edwards** used the muMATH matrix package for regression analysis. What follows is a summary of the procedure he used:

Experimental science often calls for finding a linear function that closely describes the relationships between physical variables. For example, suppose that we have collected the following 6 data points for the independent variables x_1, x_2 and the dependent variable y:

x_1	x_2	y
4	2	64
4	6	81
6	2	72
6	6	91
8	2	83
8	6	96

The problem is to determine values of the coefficients b_0, b_1 and b_2 in the equation

$$y = b_0 + b_1 x_1 + b_2 x_2 + e$$

such that the sum of the squares of the residual term, e, is minimal for the six data points. To solve the problem using muMATH, begin by entering the matrices for X and Y as follows:

```
? X: {[1, 4, 2],
      [1, 4, 6],
      [1, 6, 2],
      [1, 6, 6],
      [1, 8, 2],
      [1, 8, 6]} $

? Y: {64, 81, 72, 91, 83, 96} $

? POINT: 5$
```

The best fit coefficients are calculated by the expression:

```
? B: (X`.X) \ (X`.Y);

@: {39.33333,
    4.25,
    4.08333}
```

b_0, b_1 and b_2 are given respectively by the first, second and third components of the resultant vector. To compute the residuals for the six data points, enter the expression:

```
? E: Y - X.B;
```

```
@: {-0.5,
    0.16666,
    -1,
    1.66666,
    1.5,
    -1.83333}
```

When performed with floating-point arithmetic, this so-called method of 'normal equations' can be significantly less accurate than more sophisticated methods. However, such accuracy problems do not arise with muMATH-80 since it uses exact rational arithmetic in all calculations. Other types of regression can be done using similar matrix methods.

Using the muSIMP Pretty-printer with muSIMP-80

Milton T. Zlatic of Crestwood, Missouri pointed out that the pretty-printer listed in Newsletter #2 did not display property values when used with muSIMP-80 although it worked fine with muSIMP-79. In muSIMP-80 property values are distilled (i.e. compiled) to make them run faster and require less storage.

The following replacement definition of PRINTPROPS uses GETD to decompile and get the definition so it can be displayed.

```
FUNCTION PRINTPROPS (EX1,
   % Locals % EX2, LEX1),
  LEX1: REST (EX1),          % LEX1: PROPERTY LIST OF EX1 %
  LOOP
    WHEN ATOM (LEX1) EXIT,
    EX2: POP (LEX1),       % EX2: A PROPERTY OF EX1 %
    NEWLINE (2),
    PRINT ('PROPERTY),
    SPACES (1),
    QUOTEPRINT (EX1),
    PRINT (", "),
    PRINT (FIRST(EX2)),
    BLOCK
       WHEN REST(EX2) EQ COMPRESS (LIST (EX1, FIRST(EX2))),
        UNPARSE (2, TRUE, LIST (GETD(REST(EX2)))) EXIT,
      UNPARSE (2, TRUE, LIST (REST(EX2))),
    ENDBLOCK,
    PRINTLINE ('$),
  ENDLOOP,
ENDFUN$
```

A typographical error occurred in the definition of QUOTEPRINT that is also part of the pretty-print package listed in Newsletter #2. The **LEX2** in the eighth line of the definition should be replaced by **LEX1** and the line should read:

```
WHEN DIGIT (FIRST (LEX1))
                   ^
```

An Unrecognized 0 Problem in Content Factorization

The expression

```
FCTR (I^(1+N) + J*I^(1+N))
```

was found not to simplify to

```
(1+J)^(1+N)
```

Investigation revealed that a needed cancellation did not occur because an expression did not simplify to 0. Since using the EXPD function promotes discovery of expressions equal to 0, the problem can be resolved by making a simple change to the CANCEL1 function defined in ALGEBRA.ARI. If you have the 09/16/80 version (the version date is given on the first line of the file) of ALGEBRA.ARI, replace the line

```
WHEN ZERO (EX1: EXPON (POP(LEX2)) - EXPON (POP(LEX1))),
```

with the lines

```
EX1: EXPD (LIST ('-, EXPON(POP(LEX2)), EXPON(POP(LEX1)))),
WHEN ZERO (EX1),
```

Also change the file's version date to 01/28/82.

Finding Definite Integrals over Infinite Limits

When both INTMORE.INT and LIM.DIF have been read in, muMATH-80 has the capability to evaluate definite integrals over infinite limits. For example to integrate

$$x^{-2}$$

as x goes from 1 to positive infinity, enter the expression

```
? DEFINT (X^-2, X, 1, PINF);
@: 1
```

In order to do this, a correction must be made to the definition of the LIM function in the file LIM.DIF. If your copy of LIM.DIF is dated 07/31/81 or earlier (the version date is given on the first line of the file), change the last line of the definition from

```
LIM1 (EVSUB(EX1,INDET,EX2+INDET), #LIM),
```

to

```
WHEN INDET, LIM1(EVSUB(EX1,INDET,EX2+INDET),#LIM) EXIT,
LIM1 (EX1, #LIM),
```

and change the file's version date to 01/16/82.

SIGMA plus LIM spells Confusion

Sigma Freud from Sumwhere asks: "When both SIGMA.ALG and LIM.DIF have been read in and I enter the expression:

SIGMA (2^J, J, 0, N);

I am twice asked the sign of LN(2) and the result is unattractively expressed as a power of **e** rather than a power of 2. However, if LIM.DIF has not been read in, the desired result is returned with no questions asked. How come?"

Normally SIGMA finds the closed form sum of an expression by evaluating the expression's 'anti-difference' at the two summation limits and taking the difference. This process is analogous to computing definite integrals by evaluating their anti-derivative at the integration limits.

If the limit package has been read in, closed form sums of infinite series can also be computed. SIGMA simply uses the LIM function to evaluate the expression's anti-difference at the two summation limits. The use of LIM twice resulted in the two sign queries in your example. Since the limit package converts all exponentials to base **e**, the final result was expressed in base **e**. As in this example, the use of limits is not required for finding anti-differences at finite limits and they often gives less attractive results. To prevent their use for finding finite sums, replace

```
WHEN INTEGER(#LIM), LIM(EX4,INDET,EX3+1)-LIM(EX4,INDET,EX2) EXIT,
EVSUB(EX4,INDET,EX3+1) - EVSUB(EX4,INDET,EX2) EXIT,
```

in the SIGMA function defined in SIGMA.ALG, with the lines

```
BLOCK
  WHEN INTEGER(#LIM) AND EX2 EQ MINF, EX1: LIM(EX4,INDET,EX2) EXIT,
  EX1: EVSUB(EX4,INDET,EX2) ENDBLOCK,
BLOCK
  WHEN INTEGER(#LIM) AND EX3 EQ PINF, EX4: LIM(EX4,INDET,EX3) EXIT,
  EX4: EVSUB(EX4,INDET,EX3+1) ENDBLOCK,
WHEN INTEGER(#LIM) AND (EX2 EQ MINF OR EX3 EQ PINF), LIM(EX4-EX1) EXIT,
EX4-EX1 EXIT,
```

Change the date of the revised SIGMA.ALG source file to 03/01/82.

* * * * * * * * * * T h e m u L I S P e r * * * * * * * * * *

Using muSTAR to Create muLISP Source Files

Apparently some clarification is needed on the proper procedure for generating a pretty-printed source file using muSTAR. The problem is how to specify the function definitions and the property and variable values to be saved in the file. Section VI-A-3 of the manual states that when the muSTAR **W** command is issued, "those

functions on the property list of the file name under the indicator **FUNCTIONS**" are written to the source file.

This procedure can best be explained with the help of an example. Suppose you have defined the functions FEE, FIE, FOO, and FUM and you wish to save their definitions in the source file EXAMPLE.LIB. Begin by entering the muSTAR **P** (Edit Property) command in order to establish a property value. Following the **NAME** prompt, enter the desired file name, in this case "EXAMPLE," and following the **INDICATOR** prompt, enter the name "FUNCTIONS".

muSTAR will then clear the screen and display:

(PUTQQ EXAMPLE FUNCTIONS NIL)

Using standard muSTAR editing commands change "NIL" to the list of desired functions as follows:

(PUTQQ EXAMPLE FUNCTIONS (FEE FIE FOO FUM))

and evaluate the expression using the <CTRL-K> exit edit command. Similarly the property values and variable values to be saved can be specified by putting a list of their names on EXAMPLE's property list under the indicator "VARIABLES".

When you are ready to actually write out the file, issue the muSTAR **W** command and specify "EXAMPLE" following the **FILE NAME** prompt. Once saved, EXAMPLE.LIB can then be read into the muLISP/muSTAR system; or if more working space is necessary, it can be read into a "bare" muLISP system using an RDS command.

When the source file EXAMPLE.LIB is read in, the **FUNCTIONS** and **VARIABLES** property values of EXAMPLE will be re-established. This makes it unnecessary to repeat the above steps each time EXAMPLE.LIB is edited. Naturally, however, the list of functions and/or variables marked for saving can be enlarged or shortened as desired.

A Lexicographical Radix List Sort

Who says LISP does not make a good data processing language? The use of linked lists makes LISP a natural for the radix list sort (See Donald E. Knuth, The Art of Computer Programming, Volume 3, p. 170-178, Addison-Wesley Publishing Company, 1973). For "pipeline" or "number crunching" computers, Knuth states that "radix sorting is usually more efficient than any other known method for internal sorting."

The file **SORT.LIB** (see listing) implements the radix list sort in muLISP. The compact, efficient code illustrates the elegance of the language. Function SORTER is the top-level entry point. It is a function of one argument that should be a list of names. After the list is sorted, PRTSORT is called to display the result. Naturally PRTSORT can be replaced by a function that simply returns a sorted list that can in turn be used in further processing.

The sort is done by constructing a nested list of the names as they are sorted. Each name is unpacked and the resulting list and the current **ALST** are passed to the **SORT** function.

If the car of ALST is a name, ALST is simply a list of characters in a previously sorted name. If the the first letter of the new name character list, CLST, is not the same as the first letter ALST, ALST becomes the list containing the CLST and the old ALST, lexicographically ordered. If the first letters are the same, SORT is recursively called on the cdr of CLST and the cdr of ALST. If the cdr of CLST is empty, that fact is indicated in ALST by the inclusion of a NIL. This would occur, for instance, if "CAT" was being added to an ALST already containing "CATFISH".

If the car of ALST is not a name, the elements of ALST are lists the first element of which are characters. These lists are lexicographically sorted on these characters. SORTLST is called to find the element of ALST that matches the first character of CLST. If a match is found, SORT is called recursively on the cdr of CLST. If not, CLST is inserted in the appropriate place in ALST. The best way to fully understand the algorithm is to use the muLISP trace facility on some simple sorts.

The letters of the alphabet are included at the beginning of the file to enable ORDERP to be used for lexicographic ordering. The letter "T" has to be handled as a special case by the function ORDER since it is a primitive muLISP name.

```
% File:  SORT.LIB  (c)         01/21/82        The Soft Warehouse %

A a B b C c D d E e F f G g H h I i J j K k L l M m
N n O o P p Q q R r S s T t U u V v W w X x Y y Z z

(DEFUN SORTER (LAMBDA (LST % Local: % ALST)
  (LOOP
    ((NULL LST) (PRTSORT ALST))
    (SETQ ALST (SORT (UNPACK (POP LST)) ALST)) ) ))

(DEFUN SORT (LAMBDA (CLST ALST)
  ((NULL CLST)
    ((NULL ALST) NIL)
    (CONS NIL ALST)
  ((NULL ALST) CLST)
  ((ATOM (CAR ALST))
    ((NULL (CAR ALST)) (CONS NIL (SORT CLST (CDR ALST))) )
    ((EQ (CAR CLST) (CAR ALST))
      ((NULL (CDR ALST))
        ((NULL (CDR CLST)) ALST)
        (CONS (CAR ALST) (CONS NIL (CDR CLST))) )
      (CONS (CAR ALST) (SORT (CDR CLST) (CDR ALST))) )
    ((ORDER (CAR CLST) (CAR ALST))
      (LIST CLST ALST) )
    (LIST ALST CLST) )
  (SORTLST CLST ALST) ))
```

```
(DEFUN SORTLST (LAMBDA (CLST ALST)
  ((NULL ALST)
    ((NULL CLST) NIL)
    (LIST CLST) )
  ((EQ (CAR CLST) (CAAR ALST))
    ((NULL (CDAR ALST))
      ((NULL (CDR CLST)) ALST)
      (CONS (CONS (CAAR ALST) (CONS NIL (CDR CLST))) (CDR ALST)) )
    (CONS (CONS (CAAR ALST) (SORT (CDR CLST) (CDAR ALST))) (CDR ALST)) )
  ((ORDER (CAR CLST) (CAAR ALST))
    (CONS CLST ALST) )
  (CONS (CAR ALST) (SORTLST CLST (CDR ALST))) ))

(DEFUN PRTSORT (LAMBDA (ALST CLST)
  ((NULL ALST)
    (MAPC (REVERSE CLST) (QUOTE PRIN1)) (TERPRI) )
  ((ATOM (CAR ALST))
    ((NULL (CAR ALST))
      (PRTSORT NIL CLST) (PRTSORT (CDR ALST) CLST) )
    (PRTSORT (CDR ALST) (CONS (CAR ALST) CLST)) )
  (LOOP
    (PRTSORT (CDAR ALST) (CONS (CAAR ALST) CLST))
    (POP ALST)
    ((NULL ALST)) ) ))

(DEFUN ORDER (LAMBDA (ATM1 ATM2)
  ((EQ ATM1 T) (ORDER t ATM2))
  ((EQ ATM2 T) (ORDER ATM1 t))
  (ORDERP ATM1 ATM2) ))

(RDS)
```

```
% File DETALT.ARR  (c)        02/04/82    The Soft Warehouse %

FUNCTION MINOR (LEX2, LEX1),
  WHEN LEX2,
    WHEN LEX2 EQ LEX1, MINOR (REST (LEX2)) EXIT,
    ADJOIN (REST (POP (LEX2)), MINOR (LEX2, LEX1)) EXIT,
ENDFUN $

FUNCTION DETALT1 (LEX1, LEX2),
  WHEN EMPTY (LEX1) OR EMPTY (FIRST (LEX1)), 0 EXIT,
  WHEN EMPTY (REST (LEX2)), FIRST (FIRST (LEX2)) EXIT,
  - DETALT1 (REST (LEX1), LEX2)
    + FIRST (FIRST (LEX1)) * DETALT1 (LEX2: MINOR (LEX2, LEX1), LEX2),
ENDFUN $

FUNCTION DETALT (EX1),
  WHEN ARRAY (EX1), DETALT1 (EX1: MINOR (EX1, EX1), EX1)  EXIT,
  EX1,
ENDFUN $

RDS () $
```