N E W S L E T T E R # 7

September 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.

muMATH AND muLISP FOR 8086 AND 8088 BASED MICROCOMPUTERS

The power provided by the 16 bit, INTEL 8086 and 8088 microprocessors makes it possible to provide a greatly enhanced version of muLISP and muSIMP for these machines. A much larger work space is available for the storage of user programs and data. Naturally, execution speed has also been greatly enhanced. New features include powerful string processing functions and infinite precision integer arithmetic now limited only by available memory. **Microsoft** will soon be distributing muMATH and muLISP for the **IBM Personal Computer**. Contact The Soft Warehouse for details on availability for other computers and operating systems.

muMATH ARTICLE IN 80 MICROCOMPUTING

Bruce Douglass of A-Priori Software & Technical Writing Services writes a regular column for 80 Microcomputing entitled "Copernica Mathematica". In the June/July 1982 issue Bruce writes at length describing some application programs he has developed for muMATH. These include an integer factoring routine, a function plot routine, a cubic equation solver, and a knowledge-based system he calls "Fruit-World". It is an article well worth reading.

SOFTWARE CONTEST WINNERS

We are pleased to announce the winners of our Software Contest. **Winston Cope** wins the muMATH prize for his polynomial synthetic division program. **Stanley Schwartz** wins the muLISP prize for his SNOMED diagnostic coding program. Abstracts are printed later in this newsletter. Congratulations, Winston and Stanley. Your $100 checks are in the mail!

Solving Simultaneous Linear Algebraic Equations

Several users have wished for a function that directly solves simultaneous linear algebraic equations, thus avoiding the need to extract coefficients manually, then use matrix division. The source file LINSOLVE.MAT, as listed on page 3, accomplishes this task. It requires MATRIX.ARR and EQN.ALG as prerequisites. The function LINSOLVE is called as follows:

LINSOLVE (row or column of equations, row or column of variables)

As a convenience, any of the equations can have the "==" and right side omitted, implying a right side of 0. A consistent singular set of equations yields an answer containing forms such as ARB(1), ARB(2), etc. An inconsistent singular set of equations yields a zero-divide message and forms such as ?(1/0) in the answer. FALSE is returned if either argument is not an array, if the arrays do not have the same number of elements, or if the equations are non-linear in the specified variables. Otherwise, the solution is returned in the form

$$[var_1 == expression_1, var_2 == expression_2 \ldots],$$

where var_1 etc. are the specified variables.

For example, a typical interaction using LINSOLVE is

```
? LINSOLVE ({X + SIN(C)*Y == 3, X-2*Y}, {X, Y});

@: [X==3/(1+SIN(C)/2), Y==3/(2+SIN(C))]
```

ABSolutely Simple!

The ABS function defined in file ARITH.MUS simplifies only numeric arguments, but it is often desirable to simplify absolute values of non-numeric expressions. For example, it would be nice to have

```
ABS(ABS(4+#PI)) + ABS(-#E*X) + ABS(ATAN(X^2)) + ABS(Y^3) - ABS(Y)^3
```

simplify to

```
4 + #PI + #E*ABS(X) + ATAN(X^2)
```

Such simplifications and others are implemented in the source file ABS.ALG listed on page 4. The simplifications presume that all variables take on only real values, and arguments containing the constant #I are left as is.

```
% File:  LINSOLVE.MAT  (c)       08/21/82       The Soft Warehouse %

%        * * *  Simultaneous Linear Equations Solver  * * *        %


FUNCTION LINSOLVE (EQNS, VARS,
  % Local: % NUMNUM, DENDEN, DENNUM, PWREXPD, VARSCOPY, EQN, ROW,
    VAR, ALLVARS, TERM, RS, CF),
  WHEN ARRAY (EQNS) AND ARRAY (VARS) AND LENGTH(EQNS) EQ LENGTH(VARS),
    NUMNUM:DENDEN:DENNUM: 30,
    PWREXPD:6,
    POP (EQNS),
    POP (VARS),
    WHEN
      LOOP
        VARSCOPY: VARS,
        BLOCK
          WHEN FIRST (EQN: EVAL (POP (EQNS))) EQ '"==",
            EQN: SECOND(EQN) - THIRD(EQN) EXIT,
        ENDBLOCK,
        WHEN
          LOOP
            EQN: EQN - (TERM: EQN - EVSUB (EQN, VAR:POP(VARSCOPY), 0)),
            ALLVARS: VARS,
            TERM: TERM/VAR,
            WHEN
              LOOP
                WHEN NOT FREE (TERM,  POP(ALLVARS)) EXIT,
                WHEN EMPTY (ALLVARS), FALSE EXIT,
              ENDLOOP EXIT,
            PUSH (TERM, ROW),
            WHEN EMPTY (VARSCOPY),
              PUSH (-EQN, RS),
              PUSH (ADJOIN ('[, ROW), CF),
              FALSE EXIT,
          ENDLOOP, FALSE EXIT,
        WHEN EMPTY (EQNS) EXIT,
        ROW: FALSE,
      ENDLOOP,
        ROW: REST (ADJOIN ('{, CF) \ ADJOIN ('{, RS)),
        VARS: REVERSE (VARS),
        LOOP
          EQNS: PUSH (POP(VARS) == POP(ROW), EQNS),
          WHEN EMPTY (ROW), ADJOIN ('[, EQNS) EXIT,
        ENDLOOP EXIT EXIT,
ENDFUN $

RDS () $
```

```
% File:  ABS.ALG  (c)         08/24/82            The Soft Warehouse %

%               * * *  Absolute Value Package  * * *               %


PROPERTY #E, NONNEG, TRUE $
PROPERTY #PI, NONNEG, TRUE $

FUNCTION ABS (EX1),
  WHEN NUMBER (EX1),
    WHEN 0<EX1, EX1 EXIT,
    -EX1 EXIT,
  WHEN ATOM (EX1),
    WHEN GET (EX1, 'NONNEG), EX1 EXIT,
    WHEN GET (EX1, 'NONPOS), -EX1 EXIT,
    LIST ('ABS, EX1) EXIT,
  WHEN FREE (EX1, #I), SIMPU ('ABS, EX1) EXIT,
  LIST ('ABS, EX1),
ENDFUN $

PROPERTY ABS, ABS, FUNCTION (EX1), ABS(EX1), ENDFUN $

PROPERTY ABS, ATAN, FUNCTION (EX1), ATAN (ABS (EX1)), ENDFUN $

PROPERTY ABS, ASIN, FUNCTION (EX1), ASIN (ABS (EX1)), ENDFUN $

PROPERTY ABS, ERF, FUNCTION (EX1), ERF (ABS (EX1)), ENDFUN $

PROPERTY ABS, +, FUNCTION (EX1, EX2,
  % Local: % EX3, EX4),
  EX4: ABS(EX2),
  WHEN (EX3:ABS(EX1)) = EX1 AND EX4 = EX2
    OR ZERO (EX1+EX3) AND ZERO (EX2+EX4), EX3 + EX4 EXIT,
ENDFUN $

PROPERTY ABS, *, FUNCTION (EX1, EX2),
  ABS(EX1) * ABS(EX2),
ENDFUN $

PROPERTY ABS, ^, FUNCTION (EX1, EX2),
  WHEN INTEGER (EX2),
    WHEN EVEN (EX2), EX1^EX2 EXIT,
    ABS(EX1) ^ EX2 EXIT,
  WHEN EX1 EQ #E, #E ^ EX2 EXIT,
ENDFUN $

PROPERTY BASE, ABS, FUNCTION (EX1, EX2),
  WHEN EVEN (EX1), EX2^EX1 EXIT,
ENDFUN $

RDS () $
```

Winner of the muMATH Software Contest:

Synthetic Division of Polynomials

AUTHOR: Winston Cope, M.D.
ADDRESS: 415 7th St. South, St. Petersburg, FL, 33701
ACCESS: Public Domain

SYNDIV is a file of four functions which allow synthetic division of polynomials, written to supplement the ALGEBRA file. SYNDIV is the primary function, and its process follows the commonly used manual algorithm. Its input consists of two polynomials and their independent variable. The output is an expression consisting of the polynomial quotient plus any remainder. The name of the primary function is the same as the name of the file.

DVD, DVR, and X are passed to SYNDIV by the function call. The other names in the parameter list are local variables. QT is a list which eventually becomes the answer. QTTERM is one term of QT. FDVR is the highest-degree term of the divisor, which is constant throughout the computation. FDVD is the highest order term of the dividend, whose value changes as the computation proceeds. MD and MR are integers designating the dividend and divisor degrees respectively.

SYNDIV begins by checking for some kinds of unacceptable input. Real work begins by determining FDVR, and then a loop is entered. The exit test is performed to see if the calculation is complete, in which case the answer is given. Otherwise the function uses FINDTERM to isolate FDVD from the dividend. FDVD/FDVR then yields QTTERM, which is adjoined to QT. An updated value for DVD is calculated next, so the iteration can continue.

The helper functions are fairly straightforward. TERMPWR yields a number which is the degree of one term of a polynomial in X. MAXPWR yields a number which is the degree of a polynomial. FINDTERM extracts a term of specified degree from a polynomial.

This file is in the public domain. The author would appreciate notice and constructive criticism from anyone using it.

Judge's Comments: This is a nice package that helps fill in one of the most important limitations of muMATH. In order to give correct results, the inputs to SYNDIV must be in the form of a polynomial in the variable of interest, with all similar powers of that variable collected together into a single term. For example, use SYNDIV ((C^2-1)*X^2+X+1), (C+1)*X+1, X) rather than SYNDIV (C^2*X^2-X^2+X+1, C*X+X+1, X). It is not always easy to force muMATH expressions into this form, but successive use of EXPAND then FCTR might help.

Note too that SYNDIV is not automatically applied recursively to coefficients that are not numbers. For example, the leading term of the quotient resulting from the first example above is (C^2-1)*X/(C+1) rather than the equivalent but more compact (C-1)*X.

```
% File: SYNDIV.ALG              08/17/82              Winston Cope %

%             * * *  Synthetic Division Package  * * *             %

FUNCTION SYNDIV (DVD, DVR, X,
   % Local: %  QT, QTTERM, FR, FDVD, MD, MR),
 MD: MAXPWR (DVD, X),
 MR: MAXPWR (DVR, X),
 WHEN ZERO (MD) OR ZERO (MR) OR NOT (MR < MD), DVD / DVR EXIT,
 FDVR: FINDTERM (DVR, X, MR),
 LOOP
  WHEN MAXPWR (DVD, X) < MR, MKSUM (QT) + DVD/DVR EXIT,
  FDVD: FINDTERM (DVD, X, MAXPWR (DVD, X)),
  QTTERM: FDVD / FDVR,
  QT: ADJOIN (QTTERM, QT),
  DVD: DVD - QTTERM*DVR,
 ENDLOOP,
ENDFUN $

FUNCTION TERMPWR (EX1, X),
 BLOCK
  WHEN POWER (SECOND(EX1)),
    EX1: LIST (FIRST(EX1), THIRD(EX1), SECOND(EX1)) EXIT,
 ENDBLOCK,
 WHEN POWER (THIRD(EX1)) AND SECOND (THIRD(EX1)) = X,
  THIRD (THIRD(EX1)) EXIT,
 WHEN POWER (EX1) AND SECOND (EX1) = X, THIRD (EX1) EXIT,
 WHEN FIRST (EX1) EQ '* AND (SECOND(EX1)=X OR THIRD(EX1)=X)
   OR EX1 = X, 1 EXIT,
 0,
ENDFUN $

FUNCTION MAXPWR (EX1, X, EX2, K, KMAX),
 EX1: EX1 + 1,
 KMAX: 0,
 LOOP
  EX2: POP (EX1),
  WHEN EMPTY (EX2), KMAX EXIT,
  K: TERMPWR (EX2, X),
  BLOCK
   WHEN K>KMAX, KMAX: K EXIT,
  ENDBLOCK,
 ENDLOOP,
ENDFUN $

FUNCTION FINDTERM (EX1, X, K, EX2),
 WHEN TERMPWR (EX1, X) EQ K, EX1 EXIT,
 LOOP
  EX2: POP (EX1),
  WHEN EMPTY (EX2), FALSE EXIT,
  WHEN TERMPWR (EX2, X) EQ K AND NOT (EX2 EQ '+), EX2 EXIT,
 ENDLOOP,
ENDFUN $

RDS () $
```

Those Darn Bugs

We and some of our vigilant users have discovered some bugs and/or limitations in some of the higher level muMATH source files. The following is a list of the affected files, the revised version date, and a description of how to make the necessary changes:

1. ARRAY.ARI - 03/26/82 - The index in subscripted assignments are not evaluated, making commands like A[N]: 4 not work correctly even if N has a positive integer value. Change the definition of the subroutine UPDATE to

```
SUBROUTINE UPDATE (EX1, LEX1, EX2),
  ASSIGN (EX1, UPDATE1 (EVAL(EX1), MAPFUN ('EVAL, LEX1))),
  EX2,
ENDSUB$
```

2. MATRIX.ARR - 08/21/82 - The function IDMAT(N) generates an identity matrix of dimension N. If you feel it necessary to ensure that N is a positive integer, insert the line

```
WHEN NOT POSITIVE (EX1), FALSE EXIT,
```

   before the line

```
EX2: LIST (1),
```

   in the definition of IDMAT.

3. LIM.DIF - 03/25/82 - When using LIM with a single argument (see muMATH Reference Manual) on a function whose limit properties are unknown to the system, such as LIM(F(X)), an unnecessarily complicated answer results. Replace the line of LIM1 that reads

```
LIST ('LIM, EX1, INDET, 0),
```

   to

```
WHEN INDET, LIST ('LIM, EX1, INDET, 0) EXIT,
LIST ('LIM, EX1),
```

4. INT.DIF - 08/23/82 - In order to improve the efficiency and robustness of the "derivative divides" algorithm embodied in the function DRVDIV, change the fourth line of the definition from

```
EX5: EX1/EX4,
```

   to

```
EX5: FCTR (LIST ('/, EX1, EX4)),
```

Winner of the muLISP Software Contest:

SNOMED Diagnostic Coding Program

AUTHOR: Stanley Schwartz, M.D.
ADDRESS: Department of Pathology, The Memorial Hospital, Pawtucket, RI, 02860
ACCESS: Public Domain

SNOMED is a program to aid the Pathologist or medical secretary in coding diagnoses from the Anatomic Pathology Laboratory according to the Systematized Nomenclature of Medicine (SNOMED) which is published by the College of American Pathologists. SNOMED and its predecessor SNOP are used in many hospitals around the world in medical record departments and Pathology departments for diagnostic coding. Accurate and complete coding is necessary for valid statistics and for the retrieval of groups of cases for later scientific study.

The SNOMED.SYS program builds a dictionary of SNOMED diagnostic codes and uses that dictionary for accurate, complete, and reproducible coding. The program is written in muLISP and requires 48K to 64K of memory and one or more disk drives. You will also need a set of SNOMED code books. (Systematized Nomenclature of Medicine (SNOMED), ed 2, Skokie, IL, College of American Pathologists, 1979.)

The program works by the method of inverted dictionaries, and with minor modification it can be used as a general filing system with key-word retrieval.

SNOMED is a multiaxial and hierarchical coding system. Each diagnosis is specified by numbers along multiple axes, which are named "Topography", "Morphology", "Etiology", "Function", "Disease", "Procedure", and "Occupation". The axes are related according to the following scheme:

```
Topography + Morphology +    Etiology     + Function  =    Disease

   Lung    +  Granuloma + M. tuberculosis +   Fever   = Tuberculosis

  T-28000  +   M-44060  +     E-2001      +  F-03003  =    D-0188
```

Any given axis is optional, and for Surgical Pathology the Topography and Morphology are mostly used. The code numbers are arranged in a logical manner, with additional detail being specified with more digits. For example the code T-32 is for HEART while T-3332 is for ANTERIOR PAPILLARY MUSCLE OF RIGHT VENTRICLE.

Automatically Specifying Definitions To Be Saved While Working With muSTAR

In SWH Newsletter #6 we discussed how to manually specify the functions that muSTAR will save as a pretty-printed source file when the **W** (Write) command is issued. **David Dunthorn** of CF Systems suggested a way to automate this process by flagging functions, variables, and properties for saving when they are defined or redefined. Expanding on his idea, we came up with the following scheme.

The muSTAR Write command is modified to use the value of the global variable **SAVELIST** to specify the items to be saved. Each element of SAVELIST consists of a dotted pair: the left element is an indicator (one of the names FUNCTION, VARIABLE, PROPERTY, or FLAGGED) of the type of the item to be saved; the right element is the name of the item. This scheme allows the flexibility of having function definitions, property values, etc. intermixed in the resulting source file.

If and only if the control variable **SAVE** is "T", an item is automatically included on SAVELIST when one of the functions DEFUN, SETQQ, PUTQQ, or FLAGQQ is used to define function or assign a value. This is accomplished by calls to the function SAVELIST which adds the item to the end of SAVELIST if SAVE is "T" and the item is not already a member of SAVELIST.

The file SAVELIST.LIB gives definitions for the muSTAR functions that must be changed or added to implement the SAVELIST feature. The file can be produced using an external editor or the changes can be made interactively using the muSTAR editor itself. Once the muSTAR system has been modified, the system can be saved as a SYS file using a muLISP SAVE command.

```
% File:  SAVELIST.LIB  (c)       08/31/82       The Soft Warehouse %


(DEFUN SAVELIST (LAMBDA (NAM$ EXP$)
  ((EQ SAVE T)
    ((MEMBER (CONS NAM$ EXP$) SAVELIST))
    (SETQ SAVELIST (NCONC SAVELIST (LIST (CONS NAM$ EXP$)))) ) ))

(SETQ SAVELIST NIL)

(PUTD DEFUN (QUOTE (NLAMBDA (FUN$ EXP$)
  ((EQUAL (GETD FUN$) EXP$))
  ( ((NULL (GETD FUN$)))
    (PRIN1 "REDEFINED ")
    (PRINT FUN$) )
  (SAVELIST FUNCTION FUN$)
  (PUTD FUN$ EXP$)
  FUN$ ))) )
```

```
 (DEFUN SETQQ (NLAMBDA (NAM$ EXP$)
  (SAVELIST VARIABLE NAM$)
  (SET NAM$ EXP$)
  NAM$ ))

(DEFUN PUTQQ (NLAMBDA (NAM$ ATM$ EXP$)
  (SAVELIST PROPERTY (CONS NAM$ ATM$))
  (PUT NAM$ ATM$ EXP$)
  NAM$ ))

(DEFUN FLAGQQ (NLAMBDA (NAM$ ATM$)
  (SAVELIST FLAGGED (CONS NAM$ ATM$))
  (FLAG NAM$ ATM$)
  NAM$ ))

(DEFUN W-EXEC (LAMBDA (
    NAM$ ECHO )
  ((NULL (SETQ NAM$ (QUERY$ "FILE NAME"))))
  ((NULL (WRS (CAR NAM$) (QUOTE LIB) *DRIVE*)))
  (PRIN1 "(PUTD DEFUN ")
  (PRT-TXT (EXP-TO-TXT (GETD DEFUN) (LIST (QUOTE QUOTE))))
  (PRIN1 ")")
  (TERPRI 2)
  (PRT-TXT (DEF-TO-TXT (QUOTE (SETQQ PUTQQ FLAGQQ))))
  (TERPRI)
  (MAPC SAVELIST (QUOTE (LAMBDA (EXP$)
    (TERPRI)
    (SETQ NAM$ (POP EXP$))
    ( ((EQ NAM$ FUNCTION)
      ((TRACED EXP$)
          (UNTRACE (LIST EXP$))
          (PRT-TXT (DEF-TO-TXT (LIST EXP$)))
          (TRACE (LIST EXP$)) )
        (PRT-TXT (DEF-TO-TXT (LIST EXP$))) )
      ((EQ NAM$ VARIABLE)
      ((EQ EXP$ (EVAL EXP$)))
      (PRT-TXT (SET-TO-TXT (LIST EXP$))) )
      ((EQ NAM$ PROPERTY)
      ((GET (CAR EXP$) (CDR EXP$))
          (PRT-TXT (PUT-TO-TXT (CAR EXP$) (CDR EXP$))) ) )
      ((EQ NAM$ FLAGGED)
      ((FLAGP (CAR EXP$) (CDR EXP$))
        (PRIN2 (LIST (QUOTE FLAGQQ) (CAR EXP$) (CDR EXP$)))
        (TERPRI) ) ) ) )))
  (TERPRI)
  (PRINT "(RDS)")
  (WRS) ))

(SETQ SAVE T)

(RDS)
```