SOME REDUCE FACILITIES FOR PRETTY PRINTING SUBSCRIPTS AND FORMAL DERIVATIVES

B.J.A. Hulshof and J.A. van Hulzen

Twente University of Technology
Departments of Computer Science and Applied Mathematics
P.O. Box 217, 7500 AE Enschede
The Netherlands

Abstract

Some REDUCE facilities are presented for pretty printing formal (partial) derivatives, for array- and matrix subcripts and for operators with arbitrary arguments. The package is completely written in Standard LISP and can be considered as an addition to the output repertoire of the algebraic mode of REDUCE.

1. Introduction

When recently exploring the possibilities of an empirical algorithm, proposed by Chen and Kaup [1], to obtain conservation laws of the Benjamin-Ono equation [5], the present REDUCE facilities for vizualizing formal (partial) derivatives were felt as needlessly confusing the comprehensibility of the output. An expression like

```
4*DF(Q(X,T),X)^2*(Q(X,T)* H(DF(Q(X,T),X)) +
H(Q(X,T)* DF(Q(X,T),X)))
```

is far less pleasant than the usual notation

$$(1) \quad 4q_x^2[q.H(q_x) + H(q.q_x)]$$

In this respect MACSYMA [6] for instance is more satisfactory then REDUCE [2]. However, when consulting the sourcecode of REDUCE, we "discovered" that some slight modifications, in fact only a few hundred lines of code, already allow a much improved lay-out of the type of expressions often occuring in mathematical physics and as for example given by (1). But one must be careful with such an attempted burglary.

The internal representation of an expression in REDUCE is employed, subject to the the current output controlling flag configuration, to temporarily construct a prefix expression, which is actually used to produce the output in the usual mathematical infix notation. The line containing the leading operator of the output expression is preceded by a line in which the exponents occur. Hence the dimensioning algorithm has an upward structure, since only exponents are lifted. It creates "outputstrings", using breakpoint positions, dictated by the current linelength value. Using a subscript notation to vizualize derivatives demands an extra printline. It also requires a contraction of the "outputstrings", implying a resetting of the breakpoint positions. In addition it is elegant to implement a leftshift of the exponents of derivatives, if occuring, in the "outputstring" preceding the one containing the function symbols. We implemented these modifications. However a wish to use this subscript notation also for derivatives occuring in exponents contradicts the design philosophy of the REDUCE output routines too seriously, since it requires a two-way strategy [4]. This was beyond our intention, certainly when having an expression analysis

package [7] at our disposal, which easily allows to display or print exponents separately, after having them replaced by new identifiers. Similar facilities are implemented for matrix-and for array-subscripts and for operators with integer arguments only, which are in practice applicable as extendable arrays.

Another aspect of attempting to improve the comprehensibility of mathematical physical output which can be taken into account is the habit to frequently suppress formal arguments of functions or operators in a context dependent way. In (1) for instance H stands for the Hilbert transform operator

$$H\ q(x) = \frac{P}{\pi} \int_{-\infty}^{\infty} \frac{q(z)}{z-x}\, dz$$

and q(x,t) obeys the Benjamin-Ono equation

$$q_t + 2\ q.q_x + H(q_{xx}) = 0.$$

It is common practice to suppress the independent variables x and t, when using q, but the form on which H is operating carries essential information. Translating these habits in terms of output controlling flags, implies that it is natural to have a by default flag configuration giving automatically a subscript notation to vizualize derivatives and matrix- or array-indices, but which forces a user to tell the system for which operators (and functions are viewed as such) the arguments can be suppressed or integer arguments can be lowered. We emphasize that our modifications do not cover, notationally spoken, ambigious objects, like tensors.

In section 2 the pretty print facilities are formally introduced. Section 3 is dedicated to a sample session, illustrating the effect of the implemented output facilities. The code itself is given in [3].

2. A description of the pretty print commands.

A pretty print command, denoted by <ppcommand>, is defined as one of the alternatives following the metasymbol ::=. The alternatives are separated by the metasymbol |. Uppercase lexemes and punctation are used to describe key constructs in the command definitions, required for a succesful execution. Lowercase lexemes enclosed in <...> are names of other defining rules. So

```
<ppcommand> ::=
        ON <flag>; | OFF <flag>;|
        DOINDEX   <sidlist>;|OFFINDEX <sidlist>;|
        DONOARG   <oidlist>;|OFFNOARG <oidlist>;|
        FARG;|CLFARG;
<flag> ::= DFPRIN|NOARG
<sidlist> ::= <sid>|<sid>, <sidlist>
<sid> ::= <arrayname>|<matrixname>|<oid>
<oidlist> ::= <oid>|<oid>, <oidlist>
<oid> ::= <operatorname>
```

Once the code is installed and compiled the instruction FLOAD DFPRIN; suffices to make it operational. It is assumed that the flag NAT is on, i.e. the output is presented in infix notation. In case NAT is off all ppcommands are neglected. The initial state can be characterized by the 4-tuple PS =(ON DFPRIN, ON NOARG, OFFINDEX, OFFNOARG), i.e. derivatives on the base printline are vizualized in subscripted form and with suppressed arguments, but subscripts and argument-lists of operators not occuring in the argument-list of a DF-operator are given in the usual flat-sized way. Thus

$$f(x_1,y_{2,3})\left[\frac{\partial^5 f(x_1,y_{2,3})}{\partial^3 x_1\ \partial^2 y_{2,3}}\right]^2$$

is presented as

(2) $\quad F(X(1),Y(2,3))*F^{2}_{3X(1),2Y(2,3)}$

Variations in the actual values of PS allow to change the present pretty print state. The command OFF DFPRIN; for instance results in

$$F(X(1),Y(2,3))*DF(F(X(1),Y(2,3)),\ X(1),3,Y(2,3),2)^2$$

which is the normal REDUCE presentation of such

expressions. Observe that the influence of NOARG on the output is only effective if DFPRIN is active, i.e. NOARG is a subflag refining the impact of the DFPRIN-flag. State (2) can be restored with ON DFPRIN;. But let us now change the present state into PS= (OFF DFPRIN, .., DOINDEX, OFFNOARG) by typing in the ppcommand DOINDEX X,Y;. It gives:

(3) $F(X_1,Y_{2,3})*DF(F(X_1,Y_{2,3}),X_1,3,Y_{2,3},2)^2$

To remove the argumentlist of both occurences of F in (3) it suffices to use the command DONOARG F; which leads to

(4) $F*DF(F,X_1,3,Y_{2,3},2)^2$

Notice that the impact of this user decision is stronger than the effect of activating the subflag NOARG only, assuming DFPRIN is on, which only affects argumentlists of operators occuring as first element of a DF-argumentlist.
Now the command ON DFPRIN will give

(5) $F*F^2_{3X_1,2Y_{2,3}}$

And a combination of OFFNOARG F; and ON NOARG; finally results in

(6) $F(X_1, Y_{2,3}) * F^2_{3X_1,2Y_{2,3}}$

Hence (4) correspondents with PS=(OFF DFPRIN, .., DOINDEX, DONOARG), (5) with PS=(ON DFPRIN, .., DOINDEX, DONOARG) and (6) with PS=(ON DFPRIN, ON NOARG, DOINDEX, OFFNOARG).
The command FARG; is added to allow a user to obtain a list of all operators, extended with their argumentlists, which were previously subjected to a DONOARG-command or to ON NOARG and ON DFPRIN when occuring in a DF-argumentlist, regardless of earlier OFFNOARG-instructions. Hence in our example it results in the message
THE OPERATORS HAVE THE FOLLOWING ARGUMENTS

$F=F(X_1,Y_{2,3})$

since DOINDEX is still active. All operators mentioned in a DONOARG-command are internally stored in a list, named Farglist!*, directly after execution of the command. So a later change in the number of arguments of an operator, by adding new dimensions, is not recognized, due to the internal role of Farglist!*. It requires a clearing of Farglist!* with CLFARG; to be able to make a fresh start.

Some additional remarks:
- Composite exponents are deaf for ON DFPRIN- and DOINDEX-commands, but are subjected to DONOARG activities.
- DONOARG and DOINDEX mutually exclude each other with respect to operator argumentlists, i.e. both commands remove the possible effect of the other on these lists when being activated. An opposite instruction however does not restore the previous situation, i.e. DONOARG and DOINDEX are both cleared.

3. A sample session.

We illustrate the pretty print facilities by showing some output related to the Chen-Kaup algorithm. The idea of Chen and Kaup is to use the n-th conservation law $C_n = \int_{-\infty}^{\infty}\sigma_n dx$ to obtain the (n+1)-th by application of the empirical rule $\sigma_{n+1,x}=\delta_{n+1}-\sigma_{n,t}$, i.e. σ_{n+1} can be obtained by taking the derivative of σ_n with respect to t, followed by integration of this result with respect to x, after removal of t-dependencies using the Benjamin-Ono equation. Their empirical assumption is that $\int_{-\infty}^{\infty}\delta_{n+1}dx=0$, i.e. the integral of the sum of non-integrable terms ought to vanish, in view of the hidden existence of certain identities based on properties of the Hilbert transform operator. Hence all one has to do is sorting of and searching in rapidly increasing expressions of which the representation ought to be in some canonical form. We apply the algebraic procedures REMT and CANFORM, for removal of t dependencies and for providing a canonical form, respectively, on a simple expression and show some output related to this "testrun".

```
% --------------------
% A sample session, illustrating the possibilities
% of pretty print facilities, in combination with
% the expression analysis package [7]. The proce-
% dures REMT and CANFORM are tested using the
% expression EX.
% --------------------

*IN DFP.RED;

ALGEBRAIC PROCEDURE REMT(EXP);
BEGIN
  LET DF(Q(X,T),T)=-2*Q(X,T)*DF(Q(X,T),X)
                   -H(DF(Q(X,T),X,2)),
      DF(Q(X,T),X,T)=-2*DF(Q(X,T)*DF(Q(X,T),X),X)
                     -H(DF(Q(X,T),X,3));
  FORALL K SUCH THAT NUMBERP(K) LET
  DF(Q(X,T),X,K,T)=-2*DF(Q(X,T)*DF(Q(X,T),X),X,K)
                   -H(DF(Q(X,T),X,K+2));
  EXP:=EXP;
  FORALL K SUCH THAT NUMBERP(K)
  CLEAR DF(Q(X,T),X,K,T);
  CLEAR DF(Q(X,T),T),DF(Q(X,T),X,T);
  RETURN EXP;
END;

REMT

ALGEBRAIC PROCEDURE CANFORM(EXP);
BEGIN
  FORALL A,F SUCH THAT NUMBERP(A)
  LET H(A*F)=A*H(F);
  FORALL F,G LET
    H(F+G)=H(F)+H(G),
    H(H(F))=-F,
    H(-F)=-H(F),
    DF(H(F),X)=H(DF(F,X)),
    DF(H(F),T)=H(DF(F,T));
  EXP:=EXP;
  FORALL A,F SUCH THAT NUMBERP(A) CLEAR H(A*F);
  FORALL F,G CLEAR H(F+G),H(H(F)),DF(H(F),X),
                   DF(H(F),T),H(-F);
  RETURN EXP;
END;

CANFORM

*FLOAD SEL,DFPRIN;

FOR MORE INFORMATION TYPE ´IN SELECT.HLP$´

*OPERATOR Q,H;

*DONOARG Q;

%
% By default holds: ON DFPRIN and ON NOARG.
% DONOARG Q is used to supress Q´s arguments
% everywhere.
%

*EX:=Q(X,T)**4+3*Q(X,T)**2*H(DF(Q(X,T),X))+
*     2*DF(Q(X,T),X)**2;

       4      2            2
EX := Q  + 3*Q *H(Q ) + 2*Q
                  X       X
%
% The 3 terms building EX are seperated, named
% T(I), I=1,2,3 , and subjected to differentiation
% w.r.t T. Then the T-dependencies are removed
% with REMT, before a canonical form is produced
% with CANFORM.
% The DOINDEX facility also affects the notation
% of the lefthandsides.
%

*SETSEL;

*OPERATOR T;

*SELTRMS 1,2,3 NAME T;

*DOINDEX T;

*FOR I:=1:3 DO
*<<WRITE T(I):=T(I);
*  WRITE T(I,1):=DF(T(I),T);
*  WRITE T(I,2):=REMT(T(I,1));
*  WRITE T(I,3):=CANFORM(T(I,2))
*>>;

       4
T  := Q
 1

           3
T    := 4*Q *Q
 1,1          T

           3
T    := 4*Q *( - 2*Q*Q  - H(Q  ))
 1,2                  X      2X

           3
T    := 4*Q *( - 2*Q*Q  - H(Q  ))
 1,3                  X      2X

         2
T  := 3*Q *H(Q )
 2            X

T    := 3*Q*(Q*H  + 2*H(Q )*Q )
 2,1            T        X   T

T    := 3*Q*( - 4*Q*Q *H(Q ) + Q*H  - 2*H(Q )*
 2,2                 X    X       T        X

             H(Q  ))
                2X

T    := 3*Q*(H(Q   )*Q - 4*Q*Q *H(Q ) - 2*H(Q )*
 2,3            X,T           X    X         X

             H(Q  ))
                2X

         2
T  := 2*Q
 3       X

T    := 4*Q *Q
 3,1       X  X,T

                                          2
T    := 4*Q *( - 2*Q*Q   - H(Q  ) - 2*Q )
 3,2       X          2X      3X       X

                                         2
T    := 4*Q *( - 2*Q*Q   - H(Q  ) - 2*Q )
 3,3       X          2X      3X       X
```

```
%
% T(2,1) shows a hidden danger of the by default
% flagconfiguration ON DFPRIN, ON NOARG: The
% argument list of H in DF(H,T) is also supressed.
% T(2,2) shows that REMT does not deliver the
% desired result, which is further illustrated by
% T(2,3). These results suggest that REMT ougth to
% contain a LET-statement (and a CLEAR-statement
% to restrict its effect to the actual parameter)
% to change DF(H(EXPR),T) into H(DF(EXPR,T)),
% before the Benjamin-Ono equation can be applied.
% We single out the suspected third factor of the
% second term of EX with :
%

*T(2):=SELFACS 3 FROM T(2);

T  := H(Q )
 2       X

*FARG;
THE OPERATORS HAVE THE FOLLOWING ARGUMENTS

H=H(Q )
     X

Q=Q(X,T)

%
% FARG shows that H(DF(Q,X)) is put on Farglist!*.
% This can be needlessly confusing, since H
% operates on other expressions as well.
% Therefore the flag NOARG, for automatic
% suppression of argumentlists of (partial)
% derivatives, is turned off. Observe that
% DONOARG Q is still active, leading to the
% desired form.
%

*OFF NOARG;

*T(2,1);

3*Q*(Q*H(Q )  + 2*H(Q )*Q )
          X T         X   T

%
% This is still the wrong previous form.
%

*FORALL F,Y LET DF(H(F),Y)=H(DF(F,Y));

%
% This global definition allows to discover the
% conjectured shortcoming in REMT.
%

*T(2,1):=DF(T(2),T);

T    := H(Q   )
 2,1       X,T

*T(2,2):=REMT(T(2,1));

                                      2
T    := H( - 2*Q*Q   - H(Q  ) - 2*Q )
 2,2             2X      3X       X

*T(2,3):=CANFORM(T(2,2));

                                  2
T    :=  - 2*H(Q*Q  ) - 2*H(Q ) + Q
 2,3             2X         X     3X


%
% The conjecture is confirmed.
% To further illustrate the pretty print
% facilities the effect of earlier mentioned
% possibilities is illustrated using the just
% produced form of T(2,3), finally leading to
% the "normal" REDUCE output.
%

*OFFNOARG Q;

*ON NOARG;

*T(2,3):=T(2,3);

                                      2
T    :=  - 2*H(Q(X,T)*Q  ) - 2*H(Q ) + Q
 2,3                   2X         X     3X

*OFF NOARG;

*T(2,3):=T(2,3);

                                                  2
T    :=  - 2*H(Q(X,T)*Q(X,T)  ) - 2*H(Q(X,T) ) +
 2,3                        2X              X

       Q(X,T)
            3X

*OFF DFPRIN;

*T(2,3):=T(2,3);

T    :=  - 2*H(Q(X,T)*DF(Q(X,T),X,2)) - 2*
 2,3

                      2
       H(DF(Q(X,T),X) ) + DF(Q(X,T),X,3)

*OFFINDEX T;

*T(2,3):=T(2,3);

T(2,3) :=  - 2*H(Q(X,T)*DF(Q(X,T),X,2)) - 2*

                     2
         H(DF(Q(X,T),X) ) + DF(Q(X,T),X,3)
```

References

[1] Chen, H.H., Kaup, D.J.: Conservation laws of the Benjamin-Ono equations. J. Math. Physics 21 (1), 19-20 (January 1980).

[2] Hearn, A.C.: REDUCE User's manual. Univ. of Utah: Report UCP-19 (1973).

(Continued on page 25.)

The CSECT ALLUP (VALLUP DC V(ALLUP)) contains the source-code for the allocation process dependent on the operating system requirements.
The described modifications are only necessary for the MVS/TSO-operating system -unlike the SIEMENS BS2000-version of REDUCE (H. Kempelmann RWTH Aachen WG) which needs only the following

```
./ R 443260  443300
INLL     L       Q,CAREND          *** NEW ***
         L       A,LASTCHAR        *** NEW ***
         SR      Q,A               *** NEW ***
         LR      A,Q
         LR      14,2
         B       MKFXAT
```

substitutions to fix the length of an input line. The last step is also needed for the IBM MVS/TSO-version.

CONCLUDING REMARKS

Our extension will offer the user of the REDUCE-package some new comfortable, really interactive features within the familiar system and tries further to minimize the knowledge of JCL and the whole operating system (OS) environment. From this point of view - except for the syntax analyzing procedure which is rather primitive - we reached our goal completely.

From a system point of view we are not able to realize an extension without any modification of the original program. Neglecting the changes in the REDUCE-source because these are clear and do not produce side-effects - it remains the new OPEN of the SLISP, which seems not to be in the sense of the original REDUCE-philosophy (machine-independence as far as possible). But we hope that the obvious advantages for the user and the subroutine construction for the dynamic allocation of data sets will justify our way - further we have some ideas to realize a new version of the OPEN-section with a better portability.

Hulsof and van Hulzen continued from page 20.

[3] Hulshof, B.J.A., van Hulzen, J.A.: Some REDUCE facilities for pretty printing subscripts and formal derivatives. Twente University of Technology: Memorandum nr. INF-82-11 (November 1982).

[4] Martin, W.A.: Computer Input/Output of Mathematical Expressions. Proceedings SYMSAM II (S.R. Petrick, ed.), 78-89. New York: ACM (1971).

[5] Martini, R.,Modderkolk, E.J., van Hulzen J.A.: Some empirical investigations of an empirical algorithm to determine conservation laws of the Benjamin-Ono equation (in preparation).

[6] MATHLAB group: MACSYMA Reference Manual. M.I.T., Cambridge, Mass:Lab. of Computer Science (1977).

[7] Van Hulzen, J.A., Hulshof, B.J.A.: An expression analysis package for REDUCE, Twente University of Technology: Memorandum nr.INF-82-4 (July 1982).