ADVANCED TESTS FOR ALGORITHMS OF DIFFERENTIATION

F. Teixeira de Queiroz
Instituto Gulbenkian de Ciência - Portugal

The oldest algorithm for handling symbolic information is, perhaps, one which computes the derivative of a function. Nowadays, such an algorithm, to be a valuable tool, is never used on its own but as an element of a very large set of procedures.This set constitutes a formula manipulation system and the algorithm for differentiation certainly lies at its heart.

If, during the implementation of the system care is not taken to make sure that the procedures are consistent with each other there is a strong possibility that the system produces wrong results. With this matter in view I decided to design several easy tests to control the consistency of formula manipulation systems. With the design of this set of tests I have in mind three kinds of readers:

The first one is the traditional mathematician who supports his work with pencil and paper alone. The algorithms in the set are easily understood by him because they are based in his daily work. Therefore I hope to present him a tool with which he may obtain his results with more simplicity and more reliability. He will realize that eventually the small programs presented here can be linked up into a larger computer program.

The second type of reader I have in mind is the formula manipulation systems user. The set of tests, when adapted to the system he uses, shows him the possibilities of the system.

Finally my tests set could help the designers of formula manipulation systems to characterize and test their own differentiation algorithm. For the benefit of this type of reader I present in Appendix the ALGOL 60 description of the differentiation algorithm I use.

The test set is written in the language of my own formula manipulation system because, unfortunately, I do not have access to the most common systems in use. As a result of this circumstance the reader will find several reserved words which are not familiar to him but I hope that this fact will not be a serious obstacle to his good understanding of the tests.

The instructions which may lead to mal function ing when used in algorithms containing differentiations are the assignment, the substitution, the attribution of a name or of an expression to the derivative of some variable,and the declaration of functional dependence.

Assignment are expressions with the pattern

```
< LHS > := < RHS >;
```

where < LHS > is always an identifier and < RHS > an admissible expression. It usually contains symbolic terms as in the following example:

```
F := A + B * ( A - 1 );
```

The result of an assignment is a formula. If we have a formula containing several symbols, we can produce a new formula by substituting one or more of the symbols by expressions. These substitutions are in fact assignments of a special kind. We may, for instance,substitute in the formula F defined above T + 1 for A and 7 for B.We will then have, according to our notation, the instruction

```
G := SUBST ( F, A, T + 1, B, 7);
```

the first argument of SUBST being the identifier of the formula it deals with, followed by a list of symbols together with the corresponding expressions they are substituted by. The final result therefore is

```
G := 8 * T + 1 ;
```

Let us assume now that the manipulation system contains a facility for declaring a variable as a function of another variable. For instance, assume that the statement FUNC (F, X); means that the variable F is a function of the variable X . We may then declare that two variables X and Y are functions of two other variables A and B , introduce an assignment by which F becomes an expression containing X and Y , and we may produce a new formula containing $\frac{\partial F}{\partial A}$ and $\frac{\partial F}{\partial B}$. With these elements we are able to construct the first test:

```
'TEST 1'
FUNC (X, A);    FUNC (X, B);
FUNC (Y, A);    FUNC (Y, B);
F := X ↑ 2 + Y ↑ 2;
G := DER (F, A) ↑ 2 + DER (F, B) ↑ 2;
OUTPUT (F := F);    OUTPUT (C := G);
END;
```

In the same way we may create more complex situations by introducing several functional dependences such as, for instance F(X), B(A) and A(X) . We may thus write Test 2:

```
'TEST 2'
FUNC (F, X);    FUNC (B, A);    FUNC (A, X);
G := DER (F, X);
P := SUBST (G, F, X ↑ 2 + B);
OUTPUT (P := P);
END;
```

Consider now the case when the system includes the possibility of assigning a name or an expression to the derivative of a variable. This is a facility sometimes useful in symbolic mathematics.We use this kind of attribution when we substitute the second order differential equation

$$\frac{d^2X}{dT^2} + F \cdot X = 0$$

by the system

$$\frac{dX}{dT} = V \ , \quad \frac{dV}{dT} = -F \cdot X \ .$$

In the first of these two equations a name is assigned to the derivative and in the second an expression is assigned to the derivative.If the formula manipulation system allows assignments of this kind to be performed, new possibilities and new advanced tests become available.

The differential equation system given above is specified in our system by the statement:

```
SPECDER (T, X, V, V, - F * X);
```

in which the first symbol of the list is the independent variable and the following symbols are the assignments to the derivatives. We may use SPECDER in Test 1 and Test 2 but there are other more more powerful tests which are based on this word.

Let us consider now the homogeneous differential equation

$$Y' = \frac{X + Y}{X - Y} \ .$$

We may solve this equation by substituting the dependent and the independent variables through the substitution

$$Y = R \sin A$$
$$X = R \cos A$$

in which R and A are respectively the new dependent and the new independent variables. In order to transform the equation we take advantage of the relationship:

$$\frac{dY}{dX} = \frac{dY}{dA} \ / \ \frac{dX}{dA}$$

and, using the reserved word SPECDER , we state:

```
SPECDER (X, Y, DER (Y, A) / DER (X, A));
```

We are now able to write our test:

```
'TEST 3'
FUNC (X, A);   FUNC (Y, A);   FUNC (R, A);
SPECDER (X, Y, DER (Y, A) / DER (X, A));
F := DER (Y, X) - (X + Y) / (X - Y);
G := SUBST (F, X, R * COS(A), Y, R * SIN(A));
OUTPUT (G := G);
END;
```

We may use the above declaration SPECDER as a definition in order to compute the derivatives of Y with respect to X .

```
'TEST 4'
SPECDER (X, Y, DER (Y, A) / DER (X, A));
F0 := DER (Y, X);     OUTPUT (F0 := F0);
F1 := DER (F0, X);    OUTPUT (F1 := F1);
F2 := DER (F1, X);    OUTPUT (F2 := F2);
F3 := DER (F2, X);    OUTPUT (F3 := F3);
END;
```

The last test we present is very similar to but

much more powerful than test 4. When a differential equation does not involve explicitly the independent variable it is possible to reduce its order through a transformation of variables. If we call Y the old dependent variable and X the old independent variable we may take $P = \frac{dY}{dX}$ as the new dependent variable and Y as the new independent variable. The test consists in computing with these new variables the successive derivatives of Y with respect to X:

```
    'TEST 5'
SPECDER (X, Y, P);      DER (P, Y);
R := DER (Y, X);     OUTPUT (DF / DX := R);
R := DER (R, X);     OUTPUT (D2F/DX2 := R);
R := DER (R, X);     OUTPUT (D3F/DX3 := R);
R := DER (R, X)      OUTPUT (D4F/DX4 := R);
END;
```

APPENDIX - The procedure of differentiation which is presented next is an element of a very large system written in ALGOL 60. This system, whose listing will be supplied under request, may be considered a development of this one proposed by R. P. Van de Riet (Formula Manipulation in ALGOL 60 - Mathematisch Centrum, Amsterdam).

In this system the symbols and the numbers are stored in arrays, the formulae are represented as binarytrees with information stored in the nodes and where its elements are collected in a stack. Each element of the stack contains the information of the node and two pointers, one pointing to the LHS and the other to the RHS of the formula. The elements of the formula are stored through the procedure STORE and are analysed by the procedure TYPE 1 (TYPE % provides only the information contained in the node). The terminal nodes have pointers pointing to the arrays where the symbols and the numbers are collected. Each symbol is contained in the array IDLIST. This array has a pointer pointing to the position in the Stack where the symbol is represented as a formula.

The word SPECDER performs the loading of the SPECD array in such a way that the item SPECD [1, 0] stores the symbol of the independent variable. The word FUNC loads the three arrays DLIST, of which DLIST 1 contains the dependent variable, DLIST 2 contains the independent variable and DLIST 3 contains a pointer showing the position where the derivative is represented in the stack as a formula.

The listing of the procedure for differentiation follows below. It is simplified and it includes several auxiliar procedures which were introduced in order to simplify its understanding.

APPENDIX

```
"INTEGER" "PROCEDURE" DER (F, X); "VALUE" F, X; "INTEGER" F, X;
"BEGIN" "INTEGER" T, LHS, RHS, I, D1, D2, DRV;
  "IF" F = POINTER (X) "THEN" DRV:= 1 "ELSE"
  "BEGIN"   DRV := 0;   T := TYPE1 (F, LHS, RHS);

     "IF" T = DIFFERENTIAL "OR" T = VARIABLE "THEN"
     "BEGIN" "IF" X = SPECD [1,0] "THEN"
       "BEGIN"  "COMMENT"
          THE INDEPENDENT VARIABLE IS THE SAME AS IN SPECDER;
         "FOR" I := 1 "STEP" 1 "UNTIL" NOFSDER "DO"
         DRV := ADD (DRV, MUL (DER (F, SPECD[1,I]), SPECD[2,I]));
       "END";
       "COMMENT" THE PROGRAM TESTS ALL THE FUNCTIONAL DEPENDENCES;
       "FOR" I := 1 "STEP" 1 "UNTIL" KD "DO"
       "BEGIN"  "IF" X = DLIST2 [I]  "THEN"
          DRV := ADD ( DRV, MUL (DER (F, DLIST1[I]), DLIST3[I]));
       "END";
       "IF" T = DIFFERENTIAL "AND" DRV = 0 "THEN"
       "BEGIN"  D1 := DER (RHS, X);
       "IF" D1 "NE" 0 "THEN"  DRV:= "IF" TYPE2 (D1) =DIFFERENTIAL
       "THEN" STORE (LHS, DIFFERENTIAL, D1)  "ELSE"  DER(D1, LHS);
       "END"

     "END"  "ELSE"   "IF" OPERATION (T) "THEN"
     "BEGIN"  D1 := DER (LHS, X);   D2 := DER (RHS, X);
       DRV := "IF" T = MULTIPLICATION "THEN"
                        ADD( MUL (RHS, D1), MUL (LHS, D2))
              "ELSE" "IF" T = DIVISION  "THEN"
                        DIV ( SUB (D1, MUL(F, D2)), RHS)
              "ELSE" "IF" T = ADDITION "THEN" ADD (D1, D2)
              "ELSE" SUB (D1, D2);

     "END"  "ELSE" "IF" T = FUNCTION "OR" T = POWER N "THEN"
     "BEGIN"   D1 := DER (RHS, X);
       "IF" D1 "NE" 0 "THEN"
       DRV := "IF" T=POWER N "THEN"
                MUL ( IN(LHS), MUL (INTPOW (RHS, LHS-1), D1)
            "ELSE" "IF" LHS = EXPONENTIAL "THEN" MUL (F, D1)
            "ELSE" "IF" LHS = LOGARITHM "THEN"DIV (D1, F)
            "ELSE" "IF" LHS = SINUS "THEN" MUL (FCOS (RHS), D1)
            "ELSE" "IF" LHS = COSSINUS "THEN"
                    MUL (MINUS, MUL (FSIN (RHS), D1))
            "ELSE" "IF"  LHS = ARCTAN "THEN"
               DIV (D1, ADD (ONE, INTPOW (RHS, 2)))
            "ELSE" "IF" LHS =SQROOT "THEN"
               DIV (MUL (HALF, D1), F)

     "END"  "ELSE" "IF" T = POWER X "THEN"
       DRV := MUL(F, DER(MUL(LOG(LHS), RHS), X));
  "END";       DER := DRV;
"END" OF DER;
```