R 201

SCHOONSCHIP

A CDC 6600 program for symbolic evaluation of algebraic expressions.

M. Veltman

Abstract

A high speed program has been constructed capable of evaluating expressions of the form:

(A1 + B1 + ...) * (A2 + B2 + ...) * ... + ...

where * denotes multiplication and where A1, B1 etc. may be products of numbers, algebraic symbols, vectors, functions etc., or further expressions enclosed in brackets. The program, although rather dumb, contains several special features that are useful in working out problems in elementary particle physics: complex conjugation, spin summation, γ-matrix reduction and trace evaluation are performed on command. Facilities making possible a certain class of substitutions are present so that the answer may be expressed in terms of certain variables; or alternatively symbolic operations, like integrations if analytically possible may be performed. The output, obtainable on punched cards, is compatible with Fortran and may be used directly in any numerical program. Furthermore output may be used again as input for further processing, thus using the computer as a writing pad. This may be done also in one and the same run. Limitations on in- and output are quite generous. A typical problem, decay of a K into $\pi\mu\nu$, involving complex conjugation of a matrix element, spin-summation and trace evaluation is enclosed as an example.

Don't let any index end on E when used as dummy!

July 1967
CERN - GENEVA

Algebraic program SCHOONSCHIP

1. Introduction. The operation of this program will be explained by considering a specific, very simple example. Consider:

(A + B) * (A + B)

The program accepts expressions of this type and works out the brackets. Thus it will give, as a first result:

A**2 + A*B + B*A + B**2

(* = multiplication, ** = exponentiation). However, internally a fixed order (depending on the name of an algebraic quantity) is used for factors in a term and we get:

A**2 + A*B + A*B + B**2

As a last step, in storing the output, all terms are mutually compared, and if they are identical the numerical coefficients are added. Thus we get:

A**2 + 2*A*B + B**2

This is the basic function of the program. As one observes some rules pertinent to the particular kind of symbols involved are built in:

(i) A**1 A**1 = A**2
(ii) A*B = B*A

Other quantities that follow different rules may be used also and are described further on.

A very important facility is the substitution facility. The idea is that one may want to make changes or substitutions in the course of the calculation, or that one wants to select a particular type of term. To illustrate this suppose that we must evaluate the following integral:

$\int_0^1$dx (x + 5* A* x)* (x**2 + 3* B*x)

In such a case we know beforehand what type of integrals may appear and we could make a list:

x**3 = 0.25* x**4
x**2 = x**3/3
x**1 = 0.5* x**2

The program accepts this and will compare each producted term with the given table and make the substitutions indicated. After that the result is sorted as before.

Clearly the usefulness of the program depends on the generality of the input permitted. For instance, as a next step one may introduce vectors inside the brackets and require that products of vectors are manipulated in accordance with the rules of vector-algebra. The following section is devoted to a brief discussion of the calculational rules for the various possible quantities.

2. Rules of the game. First we will now discuss the basic rules used in the calculation. For algebraic quantities a and b we have ab = ba, and furthermore $a^m a^n = a^{m+n}$, where m and n are some numbers. Non-commuting quantities are generally called functions, whether they depend explicitly on other variables or not. As a specific example we mention γ-symbols, to be described below.

A great deal of vector- and spinor algebra has been built in, and it seems therefore useful to give first a summary of the rules of calculation employed. Some of the rules are metric-dependend (especially the rule $\delta_{\mu\mu} = r$, r = dim. of the space involved), and the metric intended is that where the metric tensor is zero, except for +1 all along the diagonal. This corresponds to the metric $p^2 = p_1^2 + p_2^2 + p_3^2 + p_4^2 = - m^2$ for a (relativistic) particle of momentum p and mass m. A vector in r-dimensional space may be denoted by p_μ , where the index μ goes from 1 to r. We assume that a dot-product is defined, i.e. with any two vectors p and q there may be associated a dotproduct (pq) that satisfies the rule (pq) = (qp). This dotproduct benaves like a commuting algebraic quantity, i.e. $(qp)^n (qp)^m = (qp)^{m+n}$, where the exponents m and n are some numbers.

As a matter of notation it is assumed that twice occurring indices imply a dotproduct. Thus:

$$p_\mu q_\mu \equiv (pq)$$

The delta-function $\delta_{\mu\nu}$ may be understood as a function forcing μ and ν to be equal:

$$p_\mu q_\nu \delta_{\mu\nu} = (pq)$$

The rule $\delta_{\mu\mu} = r$ (r = number of values taken by μ) is assumed. Furthermore $\delta_{\mu\nu}\,\delta_{\nu\lambda} = \delta_{\mu\lambda}$.

An index may take a specific value. Thus p_4 denotes the fourth component of the vector p. Of course now

$$p_4\, q_4 \neq (pq),$$

and instead quantities like p_4 and q_4 are considered to behave like ordinary algebraic quantities:

$$p_4^m\, p_4^n = p_4^{m+n}$$

The following relations hold (m and n are some numbers):

$$\begin{aligned} p_\mu \delta_{\mu m} &= p_m \quad , \text{ for instance } p_\mu \delta_{\mu 3} = p_3 \\ \delta_{mn} &= 1 \quad \text{if } m = n \text{ , for instance } \delta_{33} = 1 \\ \delta_{mn} &= 0 \quad \text{if } m \neq n \text{ , for instance } \delta_{32} = 0 \end{aligned}$$

All these rules are the well-known rules of vector-algebra. As a further development in notation we introduce the rule that a vector may occur whenever an index can occur. The meaning of the so introduced symbols is as follows (μ, ν = index, p, q = vector):

$$\begin{aligned} p_q &= (pq) \\ \delta_{\mu p} &= p_\mu \\ \delta_{pq} &= (pq) \\ \delta_{pp} &= (pp) \\ \delta_{p4} &= p_4 \end{aligned}$$

This notation avoids often the neccessity of introducing twice occurring indices.

In physical applications we will have to deal mostly with four-dimensional space. The totally antisymmetric tensor $\varepsilon_{\mu\nu\alpha\beta}$ will often occur. The essential rules are:

$\varepsilon_{\mu\nu\alpha\beta}$ is antisymmetrical under the exchange of any two indices. For instance $\varepsilon_{\mu\nu\alpha\beta} = -\varepsilon_{\nu\mu\alpha\beta}$

Further:

$$\begin{aligned} \varepsilon_{\mu\nu\alpha\beta}\,\varepsilon_{\mu\nu\alpha\beta} &= 24 \quad ; \\ \varepsilon_{\mu\nu\alpha\beta}\,\varepsilon_{\mu\nu\alpha\lambda} &= 6\,\delta_{\beta\lambda} \quad ; \\ \varepsilon_{\mu\nu\alpha\beta}\,\varepsilon_{\mu\nu\lambda\kappa} &= 2\,(\delta_{\alpha\lambda}\,\delta_{\beta\kappa} - \delta_{\alpha\kappa}\,\delta_{\beta\lambda}) \\ &= 2 \begin{Vmatrix} \delta_{\alpha\lambda} & \delta_{\beta\lambda} \\ \delta_{\alpha\kappa} & \delta_{\beta\kappa} \end{Vmatrix} \quad ; \end{aligned}$$

$$\varepsilon_{\mu\nu\alpha\beta}\,\varepsilon_{\mu\lambda\kappa\varepsilon} = \begin{Vmatrix} \delta_{\nu\lambda} & \delta_{\alpha\lambda} & \delta_{\beta\lambda} \\ \delta_{\nu\kappa} & \delta_{\alpha\kappa} & \delta_{\beta\kappa} \\ \delta_{\nu\varepsilon} & \delta_{\alpha\varepsilon} & \delta_{\beta\varepsilon} \end{Vmatrix}$$

and analogously with coefficient 1 in front of the 4 x 4 determinant if no equal indices.

Thus a product of ε's may always be reduced to zero or one ε. The program does this at the "reduction" command, or the "trace" command, to be discussed later, or after the last substitution stage.

Finally we discuss briefly γ-algebra. Consider four quantities γ^1, γ^2, γ^3 and γ^4 and the identity 1 obeying the rules:

$$\gamma^{\mu}\gamma^{\nu} = -\gamma^{\nu}\gamma^{\mu} + 2\,\delta_{\mu\nu}\,.\,1$$
$$\gamma^{\mu}1 = 1\,\gamma^{\mu} = \gamma^{\mu}$$
$$1.1 = 1$$

Define γ^5 by

$$\gamma^5 = \gamma^1\gamma^2\gamma^3\gamma^4$$

One finds:

$$\gamma^5\gamma^{\mu} = -\gamma^{\mu}\gamma^5$$
$$\gamma^5\gamma^5 = 1$$
$$\gamma^5.1 = 1.\,\gamma^5 = \gamma^5$$

These γ -symbols obey the reduction rule:

$$\gamma^{\mu}\gamma^{\nu}\gamma^{\alpha} = \delta_{\mu\nu}\gamma^{\alpha} + \delta_{\nu\alpha}\gamma^{\mu} - \delta_{\mu\alpha}\gamma^{\nu} + \varepsilon_{\mu\nu\alpha\beta}\,\gamma^5\gamma^{\beta}$$

Also, there exists a number of identities (called Chisholm identities) that may greatly reduce the labor involved in reducing a product of many γ's. A number of such identities is build in. Let S be a product of a certain number of γ's, and S_R be the same product, but with the γ's in opposite order. Thus, if $S = \gamma^{\alpha}\gamma^{\beta}\gamma^{\mu}$ then $S_R = \gamma^{\mu}\gamma^{\beta}\gamma^{\alpha}$. Furthermore brackets preceeded by Tr indicate that the trace must be taken according to the rules to be given below. Finally $\gamma^{\alpha}p_{\alpha} = \gamma^{p} = \not{p}$, where p is a vector. We will call S odd (even) if S contains an odd (even) number of γ's. We have (p^2 = dotproduct of p with itself):

$$\gamma^\mu S \gamma^\mu = -2\, S_R \quad \text{if} \quad S \ \text{odd}.$$
$$\gamma^\mu S \gamma^\mu = \mathrm{Tr}(S_R).1 - \gamma^5\, \mathrm{Tr}(\gamma^5 S_R) \quad \text{if} \quad S \ \text{even}$$
$$\not{p}\, S\, \not{p} = - p^2 S_R + \tfrac{1}{2} \not{p}\, \mathrm{Tr}(\not{p}\, S_R) + \tfrac{1}{2} \not{p} \gamma^5\, \mathrm{Tr}(\gamma^5 \not{p}\, S_R) \ \text{if } S \text{ odd}$$
$$\not{p}\, S\, \not{p} = - p^2 S_R + \tfrac{1}{2} \gamma^\nu \not{p}\, \mathrm{Tr}(\not{p} \gamma^\nu S_R) \quad \text{if} \quad S \ \text{even}.$$

These build-in identities are applied only if the command "trick" is given.

The trace of a product of γ's is given through the above quoted reduction rule (reducing any product of γ's to at most 2 γ's with or without a γ^5) in combination with the rules:

$$\mathrm{Tr}(1) = 4$$
$$\mathrm{Tr}(\gamma^\mu) = \mathrm{Tr}(\gamma^5) = 0$$
$$\mathrm{Tr}(\gamma^\mu \gamma^\nu) = 4\, \delta_{\mu\nu}$$
$$\mathrm{Tr}(\gamma^\mu \gamma^5) = \mathrm{Tr}(\gamma^5 \gamma^\mu) = 0$$
$$\mathrm{Tr}(\gamma^5 \gamma^\mu \gamma^\nu) = \mathrm{Tr}(\gamma^\mu \gamma^5 \gamma^\nu) = \mathrm{Tr}(\gamma^\mu \gamma^\nu \gamma^5) = 0$$

The trace of a product of γ's is computed according to these rules upon the command "trace".

As an additional feature γ-symbols will have another index, the so-called loop-index, to denote different sets of γ-symbols. The rules given above are pertinent to γ's within a set. γ's of different sets are behaving with respect to each other as ordinary commuting algebraic symbols.

In connection with γ-symbols we may have the spinor symbols u and $\bar{u}$. These spinor symbols are related to particles of given spin, mass and momentum. The program differentiates between spinors for spin $\frac{1}{2}$ and spinors for spin 3/2 particles. For a product of a spinor u and a spinor $\bar{u}$ of one and the same particle one may define the operation "spin sum" by (l and j are loop indices)

$$u(l,m,p)\ \bar{u}(j,m,p) = - i \not{p} + m.1 \qquad \text{for spin } \tfrac{1}{2} \text{ particle,}$$

where the γ in $\not{p}$ is given the loop-index l and where for eventually other γ's and spinors occurring anywhere else in the product the loop-index j is changed in l. The order of the terms in the product is changed: in reading the result after spin-summation the program jumps from u to $\bar{u}$ and comes back later to collect eventual factors in between. Thus, with l, j and k being loop indices the spin-sum operation changes

$$\gamma^{\alpha}_{l}\,\gamma^{\beta}_{l}\,u(l,m,p)\,\gamma^{\mu}_{k}\,\gamma^{\nu}_{k}\,\bar{u}(j,m,p)\,\gamma^{\lambda}_{j}\,\gamma^{\kappa}_{j}$$

into

$$\gamma^{\alpha}_{l}\,\gamma^{\beta}_{l}\,(-\,i\not{p}_{l} + m1_{l})\,\gamma^{\lambda}_{l}\,\gamma^{\kappa}_{l}\,\gamma^{\mu}_{k}\,\gamma^{\nu}_{k}$$

In requesting spin summation one specifies a loop index (l in the above example) and the program searches then for a spinor u with say loop-index i, and a spinor ū with the same mass and momentum variable. If found the above manipulation is performed. After this the program starts anew looking for a spinor u with loop-index i, etc. This goes on till no more such spinor u is found.

For spin 3/2 particles the spinors u and ū carry an extra index and the build in rule is:

$$u(l,\mu,m,p)\,\bar{u}(j,\nu,m,p) = \left\{1.\,\delta_{\mu\nu} - \frac{1}{3}\gamma^{\mu}\gamma^{\nu} - \frac{i}{3m}\left(\gamma^{\mu}p_{\nu} - p_{\mu}\gamma^{\nu}\right) + \frac{2}{3m^{2}}\;1.p_{\mu}\,p_{\nu}\right\}\left\{-\,i\not{p} + m.1\right\}$$

where 1 and all γ-symbols have the loop-index l.

We close this brief review by noting that an antiparticle spinor differs from a particle spinor by the mass m being replaced by -m.

3. Basic symbols, notation and definition. The following kinds of quantities are possible in the input:

(a) numericals;
(b) algebraic symbols;
(c) indices;
(d) functions;
(e) vectors;
(f) dotproducts;
(g) specific components of a vector;
(h) operators;
(i) dummies;
(j) expressions enclosed in brackets.

(a) Numericals may occur as factor in a product, as exponents of some quantity, as index of a vector to denote a specific component, and as argument of a function. As factor in a product they may be written with or without a decimal point, with or without signed exponent with or without decimal point. The exponent is indicated through the letter E and denotes the exponent of 10 with which the number has to be multiplied.

Example : the number 831 may be written as:

831   831.0   83.1E+1   83.1E+1.0
8310.0E-1.0   0.831E3

The number should never be followed by a **, symbol for exponentiation. Thus 2.4**3 is (although in principle well defined) unacceptable. But 2.4* 8.3 is o.k., * implying multiplication. Also 22.4 / 9.3E+7 is allowed, / implies division.

In all other situations numericals must be signed or unsigned integers less than 128 in magnitude. As vector-indices they must be less than 32, and unsigned. Examples:

A**107   P(4)   F(-3, A, P)
A**-107   A**(-107)

where A, P and F denote an algebraic quantity, a vector and a function respectively.

(b) Algebraic symbols are denoted by a string of one to five alphanumeric characters, the first being non-numeric, possibly followed by an exponent eventually enclosed in brackets. If no exponent present it is assumed to be +1. An algebraic symbol may occur as factor in a product, or as argument of a function. If the symbol is immediately preceeded by a / the sign of the exponent is changed.
Examples:

A   A1B**2   AB**-2

With respect to the / sign we have:

/ A = A**-1
/ AB**-2 = AB**2
/ A* A1B**2 = A**-1 * A1B**2

Here * denotes multiplication.

A symbol is defined to be an algebraic symbol if it occurs in the S-list. At most 255 different symbols are allowed. The symbol I, satisfying the rule I**2 = -1 is build in.

(c) Indices are denoted by a string of one to five alphanumeric characters, the first being non-numeric. They may occur as vector indices, as function arguments or as factors in product. In the last case they may be followed by an exponent, and are treated as algebraic symbols, see b above.

Examples:

```
P(MU3)     J**-2     / K12J**3     F(MU, NU, P)
```

where P and F denote a function and a vector respectively.

A symbol is defined to be an index if it occurs in the I-list. At most 254 different indices are allowed.

(d) Functions are denoted by a string of one to five alphanumeric characters, the first being non-numeric. They may be followed by arguments separated by comma's and enclosed in brackets. A function may appear as factor in a product or as argument of a function. Rules and possibilities concerning the arguments of a function will be given later, after the discussion of substitutions.

Examples:

```
F1K     F2(A,B,C)     F3(F2, F1K, A, B, C)
```

A symbol is defined to be a function if it occurs in the F-list. At most 223 different functions are allowed.

(e) Vectors are denoted by one or two alphanumeric characters, the first being non-numeric. A vector may occur as a factor in a product, in which case it is followed by an index, vector or number enclosed in brackets (the last two cases are really f and g, see below), or it may occur as argument of a function or other vector.

Examples:

```
P(MU)     P1(ALPHA)     F(P, P1)     P(Q)     P1(4)
```

where P, P1 and Q are vectors, MU and ALPHA indices. The rule P(-X) = - P(X) with X index or vector is build in. A symbol is defined to be a vector if it occurs in the V-list. At most 30 different vectors are allowed.

(f) Dotproducts are denoted by a symbol consisting out of two vector names separated by the letter D. They may occur as factor in a product, in which case they may be followed by an exponent and are treated as algebraic symbols. They may also appear as arguments of a function.

Examples:

PDQ   P1DP1   PDP1**3   / PDQ**3

A symbol is defined to be a dotproduct if it contains the letter D with on both sides a symbol occurring in the V-list.

(g) Specific components of a vector are denoted by a vector name followed by a positive number less than 32 enclosed in brackets. They may occur as factor in a product, in which case they may be followed by an exponent and are treated as algebraic symbols. They may also appear as arguments of a function.

Examples:

P(4)   P1(8)   P(4)**3   / P1(8)**3

A symbol is defined to be a specific component of a vector if it consists of a symbol occurring in the V-list followed by a number enclosed in brackets.

(h) An operator is a symbol occurring in the operator-list followed by an asterisk (*) and an expression enclosed in brackets on which the operation is to be performed. At the time of this writing there is only one operator, namely CONJG, denoting complex conjugation. Its doings will be described later on.

Examples:

CONJG * (A + B)   CONJG * (G(I1, P) + M)

(i) Dummies are symbols that are used in case a given function must be used several times with different symbols. The function may then be defined once, with dummy symbols at the places where the different symbols are to be used. A dummy must nevertheless be defined in S, V, F or I list, depending on what kind of quantity the dummy represents. A dummy is defined as such by occurrence on the left hand side of a substitution (to be described more detailed later on) and followed by a + sign.

Example:

ALF(J1, M+, P+) = 2*G(J1, P) + I*PDQ**3*M =

In here the expression in between = signs must be substituted whenever the function ALF with the first variable being J1 is encountered. The second and third variable are taken over and substituted in the expression inbetween = signs instead of M and P. Although P and M are dummies, they must be given in V and S list respectively, and the replacements during the calculation must be vectors or algebraic symbols respectively. If in the actual calculation an argument is some expression enclosed in brackets then the corresponding dummy must be an algebraic symbol. Thus, for the above substitution, the quantity

ALF(J1, (A+B), Q)

is computed correctly.

Dummies to be replaced by numbers must be indices.

(j) Expressions enclosed in brackets may appear as factor in a product (eventually followed by a non-negative exponent) or as argument of a function. An expression consists out of a number of terms separated by + or - signs. Each term consists of a product of factors (a) - (j).

Example,

(A*P(MU)* (C + C1) - 2*Q(4)* ALF(J1, (A+B), Q))

4. Special functions. A number of special functions are build in. Names, number of arguments expected, and meaning are:

Name	Nr of arg.	Example	Meaning
D	2	D(M,N)	$\delta_{\mu\nu}$
EPF	4	EPF(M,N,L,K)	$\varepsilon_{\mu\nu\lambda\kappa}$
G	2	G(J,M)	γ of loop J
GI	1	GI(J)	1 of loop J
G5	1	G5(J)	γ^5 of loop J
G6	1	G6(J)	$1+\gamma^5$ of loop J
G7	1	G7(J)	$1-\gamma^5$ of loop J
UG	3 or 4	UG(J,A,P)	spinor u(J,A,P) of loop J
UBG	3 or 4	UBG(J,A,P)	spinor $\bar{u}$(J,A,P) of loop J
DD	2	DD(J,K)	substitute K whenever J found.

In UG and UBG the symbols A and P stand for mass and four-momentum. For antiparticles one should use -A instead of A, P remaining as it stands. We see G7(J) = GI(J) + G5(J), which is indeed what is used in computation of traces etc. These special functions obey the rules of vector and spinor algebra given before (except DD, which was introduced for by now forgotten reasons). A spin 3/2 spinor is represented by UG(J, MU, A, P). We emphasize that A, representing a mass, must be an algebraic symbol, P a vector, and J an index. MU may be index or vector. An expression UG(J,P,A) where P and A are vector and algebraic quantity respectively leads to erratic behaviour of the program.

5. Function arguments. Function arguments may be any of the symbols (a) - (j) from section 3. These arguments may be preceeded by a - sign. Furthermore the arguments may be dummies, with the following limitations:

(i) expression enclosed in brackets may not contain dummies

(ii) a dotproduct or specific vector component may not contain dummies.

Thus the following is not allowed:

```
ALF(A+, B+) = BET((A+C), H) + 6*B =

ALF(P+, B+) = BET(PDQ) + B =
```

where A, B, C are algebraic symbols, ALF and BET functions and P, Q vectors. But, to show the point,

```
ALF(P+, B) = PDQ**-3 + B*(PDQ + P(4)) =
```

is allowed.

Some special features have been build in. An expression enclosed in brackets as function argument may be preceeded by the operational symbol CONJG*, or by -CONJG*. Example:

```
ALF(CONJG* (A + I*B), - CONJG* (A - I*C)).
```

Secondly a string of numbers (or index type dummies eventually to be replaced by numbers) separated by +, -, * or / signifying addition, subtraction, multiplication or division may also occur as argument of a function. The resulting number, at execution time, is treated modulo 128.

Example:

```
ALF(J+, K+) = BET(-2*J + K/7) + 22*B =
```

This feature is supposed to be useful in giving recurrence formula:

```
ALF(N+) = (N-1)* ALF(N-1) =
```

The usefulness is somewhat limited by the fact that expressions cannot have a negative exponent. Thus, one cannot write:

```
ALF(N+) = ALF(N-1) / (N-1) =
```

One can get around this by using two successive substitutions:

```
ALF(N+) = ALF(N-1)* BET(N-1) =
BET(N+) = 1/N =
```

6. Symbol defining lists. These lists that must preceede any given problem contain the various symbols occurring in the problem. They are punched on cards,and an S, V, I or F in the first column designates the list to be S, V, I or F list (for algebraic symbols, vectors, indices and functions respectively). The list starts in column 7, and contains the names separated by comma's. One may use up to and including column 72, and eventually continue on further cards that should have nothing punched in columns 1 - 6 and 73 - 80. The list is terminated by a period.

Examples:

```
S     A1B, C2, F3G.
I     I1, I2, I3.
V     P, P1, QQ.
F     F1, ALF, BET.
```

For indices it might be necessary to indicate the dimension of the space in which one is working (in connection with the rule $\delta_{\mu\mu} = r =$ dimension of the space). If nothing is written r = 4 is assumed. Otherwise one writes simply = r behind the index, where r is the appropriate number. Example:

```
I     I1, MU = 7, NU = 7, ALFA.
```

In S and F list properties of the symbols under complex conjugation may be given in a similar way. This is discussed in the next section.

7. Complex conjugation. In order to make complex conjugation possible the reality properties of the quantities involved must be given. The reality properties of algebraic symbols and functions are given in S and F list respectively, all other quantities are taken to be real. For algebraic symbols the notation in the S-list:

. . . . , A = I, ... implies A is imaginary;

. . . . , A = C, ... implies that A is complex. The program generates a new symbol, by appending a G, which denotes the complex conjugate of A. Here this would be AG;

. . . . , A, ... implies A is real.

In the F-list:

. . . . , F1 = I, . . . implies F1 is imaginary;

. . . . , F1 = C, . . . implies F1 is complex, with F1G as new symbol denoting the complex conjugate of F1;

. . . . , F1 = U, . . . implies that the properties of F1 under complex conjugation are undefined, and complex conjugation for such a function should be postponed till in some substitution some expression is substituted for F1. Thus, during execution time the symbol CONJG remains attached to F1 till some substitution is made.

. . . . , F1, . . . implies F is real.

Example:

S-list:

```
S      A = I, B1, B2 = C.
```

Then:

```
CONJG* (A + B1 + B2) = -A + B1 + B2G
```

With respect to functions CONJG reverses the order of the functions and performes complex conjugation as above on each function and its arguments separate. Example:

F-list:

```
F    ALF = C, ALS, F1, F2 = I, F3 = U.
```

The S-list is as above. One obtains:

```
        CONJG* (ALF(A,B1)* F1(B2) * F2(A,B2, ( A + B1))* F3(A)* F3(B1))
        = - CONJG* (F3(A)* F3(B1))* F2(-A, B2G, CONJG*(A + B1))*
            F1(B2G)* ALFG(-A, B1)
```

No "U" functions should appear as arguments of non "U" functions.

Various examples illustrating complex conjugation are given in the appendix. In the list printed out by the program the letters I, C, U are changed into certain numbers, but which have the same meaning. Thus I = 9, C = 3, U = 21. Everything following * EQUALS or * END is produced by the program.

8. Substitutions. We will now prescribe the substitution facilities. There are 39 stages in which substitutions (at most 25 per stage) can be made. They are characterized by the letter-number combinations I1, ... I7, J, J1,... J7, K, K1, ... K7, L, L1, ... L7, M, M1, ... M7.

A substitution is characterized by punching the corresponding code in column 1 and 2 of the card. In columns 3, 4, 5 a number may be punched (a comma or * in these columns is ignored); this number gives the number of times the substitution is to be inspected for. Thus, J5 5 means that the substitution is attempted at stages J5, J6, J7, K, K1. For better readability one could write J5, 5* meaning level J5, five times. In column 6 there may be a further character, specifying something concerning the substitution. In columns 7 - 72 (and, if necessary continued on subsequent cards with blanks in columns 1 - 6) follows:

(a) the identifier

(b) an = sign

(c) an expression

(d) an = sign.

The symbol punched in column 6 and the identifier determine the action taken by the program. The following summarizes the possibilities up to the time of this writing. Notation: A = algebraic symbol; P = vector; PDQ = dotproduct; J = index; P(3) = fixed component of a vector; D, D1, D2 = dummies; N = number; F, F1, F2 = functions. A dummy occurring twice in the identifier is called repetitive dummy. Example: In F1(P+, P+, A+, B+, C+) P is a repetitive dummy, A, B and C are ordinary dummies.

In an identifier exponents, functions and their arguments, and vector indices may be dummies. Thus: A**I1+* F1+ (B+, C+)* P(MU+) is allowed, but A+**2 or P+(MU) not.

Nr.	Identifier	Col. 6	Action
0	F(D1,D2,...) all dummies, no repetitive dummies	anything	Substitute once for every F
1	A=	anything but F	Substitute N times for A**N, N ⩾ 0.
2	J=	anything but F	Substitute N times for J**N, N ⩾0.
	PDQ=	anything but F	Substitute N times for PDQ**N, N ⩾0.
	P(3)=	anything but F	Substitute N times for P(3)**N, N ⩾ 0.
3	A**N=	anything	Substitute once for A**N. N maybe 0 or 1.
4	J**N=	anything	Substitute once for J**N. N maybe 0 or 1.
	PDQ**N=	anything	Substitute once for PDQ**N. N maybe 0 or 1.
	P(3)**N=	anything	Substitute once for P(3)**N.N maybe 0 or 1.
5	P(J)=	anything but D	Substitute once for P(J)
6	P(D1)=	anything but D	Substitute once for P(J), arbitrary J.
7	as 1	F	as 1, but also substituted in function arguments.
8	as 2	F	as 2, but also substituted in function arguments.
9	zero or one function, anything else. Repetitive dummies allowed.	anything but D	Substitute once if all factors are present. Example: A**D1*F(D1,D2...) is substituted if this combination occurs with D1 any number.
10	As 0, but with repetitive dummies		As 0, provided at places of repetitive dummies the same argument occurs.
11	Any other product of factors	N	Substitute once if the same combination occurs, disregard ordering of functions and possibly other functions inbetween.
12	Any other product of factors	O	As 11, not disregarding order.
13	Any other product of factors	anything but O, N	As 11, not disregarding order or inbetween functions.

Nr.	Identifier	Col. 6	Action
14	Not designated		
15	Not designated		
16	P(D)	F	Substitute in functions.
17	P(D)	D	" " dot products

With respect to 16 the following should be noted. Suppose we have:

P(J+) = EXPR(J)=

where EXPR(J) is some expression depending on J, for instance EXPR(J) = Q(J) + K(J). Suppose the function F1(A, B, P) occurs. The action taken is as follows. A new index, say M1, is created internally and placed in F1 at the place of P and in EXPR at the place of J. Next EXPR is multiplied with F1. Thus we have:

F1(A, B, P)

becomes

F1(A, B, M1)* EXPR (M1)

Often one will use identity type 16 to express the fact that some vector equals the sum of a number of other vectors. Care should be taken that such vectors occur only in functions that are linear in the particular argument. Examples of such linear functions are δ and ε (i.e D and EPF), and γ (i.e. G) with respect to its second argument. But UG and UBG are in general not to be considered linear except when the mass involved is zero. Thus if UG and UBG occur substitution 16 should be performed after spin summation, but before trace evaluation (in view of the fact that trace-evaluation generates dotproducts, and the substitution 16 is not performed on vectors occurring in dotproducts). The appendix gives some examples of substitutions.

At this point we must make an important remark. Any deeper level bracketing leads to a higher level substitution. For instance the expression:

(A+ B* (C+D))

leads to a newly created symbol, $\$$ 777, and a new substitution:

(A+ B* $\$$ 777)

I1 $\$$ 777 = C + D =

Other example:

A + B* (C* (F + H) + K)

leads to

```
        (A + B* $ 777)
I1      $ 777 = C* $ 776 + K =
I2      $ 776 = F + H =
```

In this type of manipulation dummies are carried along correctly (with the limitation that no expressions occurring as function arguments contain dummies).

Thus, if deeper level bracketing occurs one must leave space in the identity levels. For instance, a substitution H = 3* A + B = should in the last example be placed at I3 or higher, as H appears only at level I2.

9. Commands. A number of useful commands is build in. As with substitutions columns 1 - 5 are used to indicate the stage at which the command is to be followed up. Reduction and trace evaluation need two stages, if the Chisholm identities are to be applied more stages are needed. In the last case one should indicate how many stages can be used, at least 3. Typically 5 stages will be allright, in that case the Chisholm identities are applied 4 successive times. The commands available are (J1, J2 etc. are loop indices):

REDUC, J1, J2, ... = Reduce product of γ's of loops J1, J2 etc. to at most two γ's and eventually a γ^5.

TRACE, J1, J2, ... = Take trace of γ's of loops J1, J2, ...

TRICK, I1, I2, ..., TRACE, J1, J2, ... = Reduce product of γ's of loops I1, I2, ..., take trace of γ's of loop J1, J2, ... Use Chisholm identities as much as possible within the allowed number of stages.

SPIN1, J1, J2, ... = Perform spin summation over loops J1, J2, ...

It should be remembered that while doing spin summation loop indices might be changed. For instance the combination UG(J1, M, P)*UBG(J2, M, P) leads to change of J2 into J1 everywhere. Thus J2 needs not to be mentioned in the list. However, no harm results from J2 being in the list.

EVENX, AF, 1,2,4,BF,2 = Recognize AF as a function even in arguments nr. 1,2 and 4, and BF even in argument 2. Thus remove eventual -signs of these arguments. Example: AF(A1, -A2, -A3, P1, B1) goes over into AF(A1, A2, -A3, P1, B1)

ODDXX,AF,1,2,4,BF,2 = Same as above, only for every -sign removed the function AF gets a -sign: AF(A1, -A2, -A3, P1, B1) goes over into -AF(A1, A2, -A3, P1, B1)

REPLA,A1,A2,AF,1,2,4,BF,2 = If arg. nrs. 1,2 or 4 of AF or 2 of BF contain A1 replace it by A4.

ASYMX,AF,1,2,3,BF,2,3 = Recognize that AF is antisymmetric in arguments 1,2, and 4. Set them in a certain (internally well-defined) order, while eventually changing sign. Same for BF. Example: AF(A1,B1,C1,D1) goes over into - AF(C1,B1,A1,D1). Also AF(C1,D1,A1,B1) goes over into -AF(C1,B1,A1,D1). If two arguments are equal the term is made zero.

SYMXX,AF,1,2,3,BF,2,3 = Same as ASYMX, only no sign charge.

The following two commands are introduced for matrix manipulation. Essentially the idea is to regroup a product of matrices, like $A_{ij}\ B_{ki}\ C_{jk}$ such that equal indices follow each other. Thus we want the result to be $A_{ij}\ C_{jk}\ B_{ki}$. For such purposes there is the command:

ORDER, A, 1, I1, I2, I3 =

This means: search function A. Start with the index which is following (if there is no index following start with the preceeding) argument number 1. Find a function which has the same index as argument, and place it behind function A. Get as new index the index following the previous one and proceed as above. Go on till no more index found, or till one arrives at a function which is already part of the chain. Next search index I1, and if found as function argument start a new chain with that function. Etc.

If, on searching the function A it is found that A is already part of a chain the program skips that A and goes on searching. This can happen if the same command is given twice at the <u>same</u> stage. Thus:

```
A(I1, I2)* B(I3, I1)* C(I2, I3)* A(J1, J2)*
B1(J3,J1)* C1(J2, J3)
```

needs twice the same command at the same stage

```
ORDER, A, 1 =
ORDER, A, 1 =
```

to give:

```
A(I1, I2)* C(I2, I3)* B(I3, I1)* A(J1, J2)* C1(J2, J3)* B1(J3, J1)
```

The command

```
ORDEI, I1, I2, I3 =
```

leads to the same work, only the program first finds index I1, and starts with the function containing I1. This might give rise to "anti" ordering, i.e.

```
A(I1, I2)* C(I2, I3)* B(I3, I1)
```

is on the command

```
ORDEI, I2 =
```

rearranged in

```
A(I1, I2)* B(I3, I1)* C(I2, I3)
```

The following are two commands meant for manipulation of orthogonal vectors. The command:

```
ORTHG, Q1, P1, P2, P3 =
```

results in that the dotproducts Q1DP1, Q1DP2 and Q1DP3 are set to zero. The command:

```
ORTHN, P1, P2, P3 =
```

results in P1DP2, P1DP3 and P2DP3 are set to zero while P1DP1, P2DP2 and P3DP3 are set to one.

The following command is intended for selection of terms. First there must be an A-list, consisting of an A in the first column, and in columns 7 - 72 follow names (vector, index, algebraic symbol, function) followed by an = sign and a signed or unsigned number less than 2^{17} in magnitude. All are separated by comma's and the list is terminated by a period. No dotproducts or fixed components of a vector should occur in the list.

Example:

```
A     A1 = 3, B2 = -5, AF = 7, P = 5.
```

The command

```
ASYMP, 10 =
```

leads to the following action. Every term is inspected, and an "asymptotic" number is constructed. This number is obtained by adding 3 for every occurrence of A1, -3 for A1**-1, -5 for B2 etc. Thus A1**7 adds 21. Function arguments are not counted. Dotproducts and fixed components of a vector are counted with respect to the vectors involved. Thus PDP contributes 10. If the so constructed asymptotic number is equal or larger than the number in the ASYMP command (here 10) the term is kept, and otherwise set to zero.

In general at a given stage no more than one command should be given. Also substitutions are not to be given together with a command at one stage. This holds also for substitutions arising from deeper level brackets.

10. Miscellaneous remarks. A card with an asterisk (*) in column one indicates that the whole problem is read, and that evaluation can start. After the problem is evaluated the action taken depends on the further contents of the card with the *. If

```
END  in column 7, 8, 9
```

the program writes output, punches eventually cards, rewinds tapes etc. and stops.
If:

```
YEP in column 7, 8, 9
```

the program writes the output on a certain tape and uses that subsequently as input, applying identities following the YEP card.*
A YEP command leaves S, F, I, V and A lists unchanged, but in general a new B-list (see section 11) must be given. Also the N card looses validity. If anything else the program writes and punches (when requested for) and next considers the cards following the card with an * as a new problem to be attacked.

If the input for any problem consists of the output of a foregoing run the cards in question must be preceeded by a card having an R in the first column. I, S, F, V and A lists, as well as identities and the B list (discussed in section 11) must be given before the R card. The symbol CONJG must not appear in the cards

* The identities are followed by a card with an *

following the R card. The whole should be terminated by a card with an* in the first column.

11. Output instructions. In the output vectors and functions are factored out. If one wants to factorize any other quantity, then this quantity must be mentioned in the B list (B in first column, names separated by comma's in columns 7 - 72, terminated by a period. At most 25 names may be given). This B list must be after S, V, F and I lists. A proper factorization facilitates the ordering and comparison of terms in the output, and for problems producing a large output one should take care to obtain roughly a situation with M = N where, given the output

F1 *(A1 + B1 + ...) + F2 *(A2 + B2 + ...) + ...

the number N is the number of factors F1, F2, etc, while M is the average number of terms inside brackets. Shortly: about as many brackets as terms inside brackets.

Further control on the output is achieved by punching symbols in columns 73 - 80 of the card preceeding a given problem. This will usually be a card with an * in the first column. The possibilities are:

nothing anywhere: print both input and result, do not punch cards
/ in col. 73 : punch cards of output.
/ in col. 74 : do not print output.
s̸ in col. 74 : print intermediate result in case of a YEP.
/ in col. 75 : do not print input.
s̸ in col. 76 : print working out of brackets.

Next, a card with an N in the first column, and a number in col. 7 and 8 followed by a period instructs the program to print floating point numbers with as many decimals as the number specifies.*

In the output terms are arranged such that terms containing factors quoted first in the S list are printed first.

* If not specified it is taken to be 5.

12. Ordering cards. The program, presumably on binary cards, is followed by a card *DATA. Next comes a card specifying tape numbers:

Col. 9, 10 Output tape. (= 2 for the CERN system)

Col. 19, 20 If non zero, output is printed on this tape in the same format as punched cards.

Col. 29, 30
Col. 39, 40 } 2 tapes used for intermediate storage in case of large amounts of output.

Col. 49, 50 Tape used for storing output to be used as input in case of a YEP command.

Columns 72 - 80 contain output regulating symbols, see section 11.

Next the first problem is given. First, there must be the various lists : S, V, I, F and B and eventual A and N cards (see section 9 and 11). Secondly the main expression is given on cards with blanks in columns 1 - 6 (and 72 - 80 which are ignored anyway). This main expression must consist of expressions in brackets separated by *, + or - signs. Next follow the substitutions and commands, and the whole is terminated by a card having an * in col. 1. After this card, if it does not contain END or YEP in col. 7, 8, 9 further problems may be given. The last card of the pack should be a card with an * in col. 1 and END in 7, 8, 9.

13. Some further comments. A limited amount of diagnostics is build in, for some of the more obvious errors. Also, the program prints out the various name lists, and in case the same name has been used for different quantities it will appear in a list "Confused" . One should be quite careful to avoid that that for instance algebraic symbols appear at places where vector symbols are expected, as this may result in the program stopping in the midst of the calculation without printing warning or reason. Or, as may often happen in such cases, the message "trouble with indices" is given. For instance D(M,A) where M is an index and A an algebraic symbol leads to disaster when multiplied by say P(M), P being a vector.

Often it is quite difficult to see how much computer time a given problem may take. The time needed is more or less proportional to the number of terms arriving at the output before sorting. Roughly 5 minutes must be counted per 100.000 terms. Quite often simple tricks in the input can greatly influence the number of terms produced; also doing halfway the problem a YEP may help, as a YEP implies sorting of the terms before going on (see example on $K \to \pi\mu\nu$ in the appendix). Be careful that at the point where the YEP is done all complex conjugation is performed, i.e. that all "U" type functions have been substituted. Also a new B list must be given after a YEP.

If, while reading the problem the program encounters a symbol not defined in any list a guess is made taking into account the way the symbol appears. To give the idea: if a new symbol appears as argument of a function it is

(i) assumed to be a specific component of a vector if the symbol is followed by a bracket (;

else (ii) assumed to be a dotproduct if a D appears in the name, and not as first or last letter;

else (iii) assumed to be an index if the name starts with I, J, K, L, M or N;

else (iv) assumed to be an algebraic symbol.

It must be remembered that certain symbols are built in. Thus do not call an algebraic symbol D or G, as there are already functions by that name. Also I should not be used as index, because I is reserved for the quantity obeying I**2 = -1.

Note that all output is terminated by + 0. This + 0. should be kept in using output of a previous run as input.

14. Internal representation. Internally symbols are represented by 12-bit quantities. The heading 4 bits specify what kind of symbol, the last 8 specify the name of the symbol.

heading (digital)	kind	example (in octal)	
0000	dummy	0007	(dummy nr. 7)
0001	index	0403	(index nr. 3)
0010	vector	1005	(vector nr. 5)
0011	operator	1420	CONJG
0100	algebraic symbol	2010	(symbol nr. 8)
0101	expr. in brackets	2513	(expr. nr. 75)
0110	function	3023	(function nr. 19)
0110	function end	3000	
0111	number	3405	(5)
10**	component of a vector The 10 is followed by a 5-bit vector name and a 5-bit number.		
11**	dotproduct. The 11 is followed by two 5-bit vector-names.		

Functions appearing as argument of a function have the vector heading, and then the function name with 32 added on. Thus 1041 is function nr. 1 appearing as argument of some other function.

Appendix. In the following some comments on the examples given on pages A1 - A25 will be given.

The general structure of the printed output is as follows. First the input is listed, up to and including the card containing an asterisk in the first column. The text after that is produced by the program: lists of the various symbols, result of the calculation, and finally some statistics on the problem. The running time is in milli-minutes. The number of terms actually printed equals T - E - C, where T = number of terms, E = number of equal terms and C = number of cancellations. The number of records indicates how many records have been written on tapes for storing intermediate results (not including the intermediate results in case of a YEP command).

A1, A2 Show complex conjugation of algebraic symbols.

A3 Demonstrates that twice conjugated gives the original expression. Note the ordering of the final result in accordance with the ordering in the S-list.

A4 Demonstrates complex conjugation of a product of functions. Note that this output cannot be used as input again because of the occurrence of the CONJG symbol.

A5 Demonstrates use of dummies, and the occurrence of an expression in brackets and minus signs in function arguments. Notice that -PP inserted for the dummy P in for instance P(4) gives -PP(4). If we had written P(4)**2 the result would have been PP(4)**2.

A6, A7 Complex conjugation of function and its arguments.

A8 Example of identity to be used in case of spin summation for spin 1 particles. Shows also the use of the / sign. There is no deep reason for writing M/M**3 instead of /M**2.

A9-A16 Demonstrates the use of some substitution types.

A15 Note how a seemingly innocent expression gives rise to a huge number of terms (40.000, for some reason that is no more relevant the count was 4 too high). That the time taken is still relatively small is because the output store contained never many terms, so that comparison of a new term with the already present ones could go very rapidly.

A17-A20 Demonstrates application on a problem encountered in elementary particle physics. Notice that in DIA the indices MU, NU had to be taken as dummies because otherwise DIA * CONJG * (DIA) would have given rise to terms containing 4 times the identical index.

On page A 18 the program did not print out the card * END, instead printed an R card that is preceeding the input coming from the intermediate tape (which was written there in response to the YEP command). This intermediate input - output is not listed, and after the R-card the outcome of the whole problem starts.

If this output is to be used in a Fortran program it might be worthwhile to introduce certain abbreviations with the help of substitutions. For instance some obvious ones:

```
M**2 = M2 =
M**4 = M4 =
```

etc. Also one could have given directly the numerical values of the various masses. For instance to remove M one could use the substitutions :

```
M = 105.6 =
M**-1 = 1./ 105.6 =
M**-2 = 1./ 105.6 / 105.6 =
```

Notice that negative powers have to be listed separately, as the first substitution is only performed on positive powers of M.

The time taken for this problem was about 0.9 minutes. The calculation could have been speeded up if we had used the vector QQ instead of Q + QP in the input, giving the necessary substitutions to remove QQ again after the YEP, like:

```
KDQQ = KDQ + KDQP
```

etc. Of course, substitutions like QPDQP = 0 have then to be given again, after the removal of QQ.

A21-A25 Demonstrates ASYMP, ORDER, ORDEI, ORTHG and ORTHN.

```
S       A=I,B1,B2=C,BC2,B3,B4=I,B5=C,B55.
I       I1.
F       ALF=C,ALS,F1,F2=I,F3=U,H1,H2=I,H3=C,H33.
V       P,Q,PP,QQ,R.
        (CONJG*(A)+A1*CONJG*(B1)+A2*CONJG*(B2)+A3*CONJG*(BC2)+A4*
            CONJG*(B4)+A5*CONJG*(B5)+A6*CONJG*(B55))
*       EQUALS

SYMBOLS             I=9, A=9, B1, B2=3, B2G, BC2, B3, B4=9, B5=3, B5G, B55, A1,
                    A2, A3, A4, A5, A6.

INDICES             I1.

VECTORS             P, Q, PP, QQ, R.

FUNCTIONS           D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3, UBG, DD, ALF=3, ALFG,
                    ALS, F1, F2=9, F3=21, H1, H2=9, H3=3, H3G, H33.



 - A + B1*A1 + B2G*A2 + BC2*A3 - B4*A4 + B5G*A5 + B55*A6



 +0.

RUNNING TIME              7
NUMBER OF TERMS           7
EQUAL TERMS               0
CANCELLATIONS             0
RECORDS                   0
```

A1

```
S       A=I,B1,B2=C,BC2,B3,B4=I,B5=C,B55.
I       I1.
F       ALF=C,ALS,F1,F2=I,F3=U,H1,H2=I,H3=C,H33.
V       P,Q,PP,QQ,R.
        (CONJG*(A+A1*B1+A2*B2+A3*BC2+A4*B4+A5*B5+A6*B55))
*       EQUALS

SYMBOLS         I=9, A=9, B1, B2=3, B2G, BC2, B3, B4=9, B5=3, B5G, B55, A1,
                A2, A3, A4, A5, A6.

INDICES         I1.

VECTORS         P, Q, PP, QQ, R.

FUNCTIONS       D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3, UBG, DD, ALF=3, ALFG,
                ALS, F1, F2=9, F3=21, H1, H2=9, H3=3, H3G, H33.




 - A + B1*A1 + B2G*A2 + BC2*A3 - B4*A4 + B5G*A5 + B55*A6



 +0.

RUNNING TIME            6
NUMBER OF TERMS         7
EQUAL TERMS             0
CANCELLATIONS           0
RECORDS                 0
```

```
S       A=I,B1,B2=C,BC2,B3,B4=I,B5=C,B55.
I       I1.
F       ALF=C,ALS,F1,F2=1,F3=U,H1,H2=I,H3=C,H33.
V       P,Q,PP,QQ,R.
         (CONJG*(CONJG*(A+A1*B1+A2*B2+A3*BC2+A4*B4+A5*B5+A6*B55)))
*       EQUALS

SYMBOLS           I=9, A=9, B1, B2=3, B2G, BC2, B3, B4=9, B5=3, B5G, B55, A1,
                  A2, A3, A4, A5, A6.

INDICES           I1.

VECTORS           P, Q, PP, QQ, R.

FUNCTIONS         D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3, UBG, DD, ALF=3, ALFG,
                  ALS, F1, F2=9, F3=21, H1, H2=9, H3=3, H3G, H33.




 + A + B1*A1 + B2*A2 + BC2*A3 + B4*A4 + B5*A5 + B55*A6


 +0.

RUNNING TIME              7
NUMBER OF TERMS           7
EQUAL TERMS               0
CANCELLATIONS             0
RECORDS                   0
```

A3

```
S       A=1,B1,B2=C,BC2,B3,B4=I,B5=C,B55.
I       I1.
F       ALF=C,ALS,F1,F2=I,F3=U,H1,H2=I,H3=C,H33.
V       P,Q,PP,QQ,R.
        (CONJG*(ALF(A,B1,B2,BC2,I1,P,QDP,P(4),H1,H2,H3,H33)*
                 ALS(A,B1,B2,BC2,I1,P,QDP,P(4),H1,H2,H3,H33)*
                 F1 (A,B1,B2,BC2,I1,P,QDP,P(4),H1,H2,H3,H33)*
                 F2 (A,B1,B2,BC2,I1,P,QDP,P(4),H1,H2,H3,H33)*
                 F3 (A,B1,B2,BC2,I1,P,QDP,P(4),H1,H2,H3,H33)*
                 F3 (  B1,B2,BC2,I1,P,QDP,P(4),H1,H2,H3,H33)*
                 H1 (  B1,B2,BC2,I1,P,QDP,P(4),H1,H2,H3,H33)*
                 H3 (  B1,B2,BC2,I1,P,QDP,P(4),H1,H2,H3,H33)  )  )
*       EQUALS
```

```
SYMBOLS         I=9, A=9, B1, B2=3, B2G, BC2, B3, B4=9, B5=3, B5G, B55.

INDICES         I1.

VECTORS         P, Q, PP, QQ, R.

FUNCTIONS       D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3, UBG, DD, ALF=3, ALFG,
                ALS, F1, F2=9, F3=21, H1, H2=9, H3=3, H3G, H33.
```

```
- H3G(B1,B2G,BC2,I1,P,PDQ,P(4),H1,-H2,H3G,H33)*H1(B1,B2G,BC2,I1,P,PDQ,P(4),H1,-H2,H3G,H33)

*CONJG*(F3(A,B1,B2,BC2,I1,P,PDQ,P(4),H1,H2,H3,H33)*F3(B1,B2,BC2,I1,P,PDQ,P(4),H1,H2,H3,H33)

)*F2(-A,B1,B2G,BC2,I1,P,PDQ,P(4),H1,-H2,H3G,H33)*F1(-A,B1,B2G,BC2,I1,P,PDQ,P(4),H1,-H2,H3G,H33)

*ALS(-A,B1,B2G,BC2,I1,P,PDQ,P(4),H1,-H2,H3G,H33)*ALFG(-A,B1,B2G,BC2,I1,P,PDQ,P(4),H1,-H2,H3G,H33)

+ 0.
```

A4

```
S       A=I,B1,B2=C,BC2,B3,B4=I,B5=C,B55.
I       I1.
F       ALF=C,ALS,F1,F2=I,F3=U,H1,H2=I,H3=C,H33.
V       P,Q,PP,QQ,R.
         (ALF((A1+B2),-A,-PP,QQ))
J       ALF(A+,B+,P+,Q+)=A**3+B+3*A2*B**5+P(4)+PDQ+D(P,Q)*A3+PDR+QDR=
*       EQUALS

SYMBOLS          I=9, A=9, B1, B2=3, B2G, BC2, B3, B4=9, B5=3, B5G, B55, A1,
                 A2, A3.

INDICES          I1.

VECTORS          P, Q, PP, QQ, R.

FUNCTIONS        D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3, UBG, DD, ALF=3, ALFG,
                 ALS, F1, F2=9, F3=21, H1, H2=9, H3=3, H3G, H33.
```

```
 - 3.*A**5*A2 - A + 3.*B2*A1**2 + 3.*B2**2*A1 + B2**3 + A1**3 - A3*PPDQQ - PP(4)

 - PPDQQ - PPDR + QQDR


 +0.

RUNNING TIME              8
NUMBER OF TERMS          15
EQUAL TERMS               4
CANCELLATIONS             0
RECORDS                   0
```

A5

```
S       A=I,B1,B2=C,BC2,B3,B4=I,B5=C,B55.
I       I1.
F       ALF=C,ALS,F1,F2=I,F3=U,H1,H2=I,H3=C,H33.
V       P,Q,PP.QQ,R.
        (CONJG*(ALF(-(A1+B2),-A,-PP,QQ)) )
J       ALFG(A+,B+,P+,Q+)=A**3+B+3*A2*B**5+P(4)+PDQ+D(P,Q)*A3+PDR+QDR=
*       EQUALS

SYMBOLS           I=9, A=9, B1, B2=3, B2G, BC2, B3, B4=9, B5=3, B5G, B55, A1,
                  A2, A3.

INDICES           I1.

VECTORS           P, Q, PP, QQ, R.

FUNCTIONS         D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3, UBG, DD, ALF=3, ALFG,
                  ALS, F1, F2=9, F3=21, H1, H2=9, H3=3, H3G, H33.




 + 3.*A**5*A2 + A - 3.*B2G*A1**2 - 3.*B2G**2*A1 - B2G**3 - A1**3 - A3*PPDQQ - PP(4)

 - PPDQQ - PPDR + QQDR



 +0.

RUNNING TIME              7
NUMBER OF TERMS          15
EQUAL TERMS               4
CANCELLATIONS             [illegible]
RECORDS                   [illegible]
```

```
S       A=I,B1,B2=C,BC2,B3,B4=I,B5=C,B55.
I       I1.
F       ALF=C,ALS,F1,F2=I,F3=U,H1,H2=I,H3=C,H33.
V       P,Q,PP,QQ,R.
        (CONJG*(ALF(-CONJG*(A1+B2),-A,-PP,QQ)) )
J       ALFG(A+,B+,P+,Q+)=A**3+B+3*A2*B**5+P(4)+PDQ+D(P,Q)*A3+PDR+QDR=
*       END

SYMBOLS          I=9, A=9, B1, B2=3, B2G, BC2, B3, B4=9, B5=3, B5G, B55, A1,
                 A2, A3.

INDICES          I1.

VECTORS          P, Q, PP, QQ, R.

FUNCTIONS        D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3, UBG, DD, ALF=3, ALFG,
                 ALS, F1, F2=9, F3=21, H1, H2=9, H3=3, H3G, H33.




 + 3.*A**5*A2 + A - 3.*B2*A1**2 - 3.*B2**2*A1 - B2**3 - A1**3 - A3*PPDQQ - PP(4)

 - PPDQQ - PPDR + QQDR



 +0.

RUNNING TIME           6
NUMBER OF TERMS       15
EQUAL TERMS            4
CANCELLATIONS          0
RECORDS                0
```

```
F       EG=C.
V       P.
S       M.
I       MU,NU.
        (EG(MU,M,P)*EGG(NU,M,P))
L     N EG(MU+,M+,P+)*EGG(NU*,M+,P+)=D(MU,NU)+M*P(MU)*P(NU)/M**3=
*       EOUALS

SYMROLS            I=9, M,

INDICES            MU, NU.

VECTORS            P.

FUNCTIONS          D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3, UBG, DD, EG=3, EGG.




+ D(NU,MU)


+ P(NU)*P(MU)

* ( M**(-2) )


 +0.

RUNNING TIME              7
NUMBER OF TERMS           2
EQUAL TERMS               0
CANCELLATIONS             0
RECORDS                   0
```

A8

```
F       F1,F2,F3,F4,F5,F6,F7.
V       P,Q,
        (F1(A,B,C)*F2(A,B,C)*F3(A,B,C)*F1(A,B,C))*(A**3*PDQ**2*P(MU))
K       F1(D1+,D2+,C)=F5(D1,D2)=
K1    N F2(D1+,D2+,C)*F4(D1+,D2+,C)=F6(D1,D2)=
L       A**D1+=C**D1=
*       EQUALS

SYMBOLS        I=9, A, B, C.

INDICES        MU.

VECTORS        P, Q.

FUNCTIONS      D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3, UBG, DD, F1, F2, F3,
               F4, F5, F6, F7.




+ P(MU)*F5(A,B)*F2(A,B,C)*F3(A,B,C)*F5(A,B)

* ( C**3*PDQ**2 )


 +0.

RUNNING TIME          5
NUMBER OF TERMS       1
EQUAL TERMS           0
CANCELLATIONS         0
RECORDS               0
```

```
F       F1,F2,F3,F4,F5,F6,F7.
V       P,Q,
        (F1(A,B,C)*F2(A,B,C)*F3(A,B,C)*F4(A,B,C))*(A**3*PDQ**2*P(MU))
K1     N F2(D1+,D2+,C)*F4(D1+,D2+,C)=F6(D1,D2)=
*       EQUALS

SYMBOLS          I=9, A, B, C.

INDICES          MU.

VECTORS          P, Q.

FUNCTIONS        D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3, UBG, DD, F1, F2, F3,
                 F4, F5, F6, F7.




+ P(MU)*F1(A,B,C)*F3(A,B,C)*F6(A,B)

* ( A**3*PDQ**2 )


 +0.

RUNNING TIME          15
NUMBER OF TERMS        1
EQUAL TERMS            0
CANCELLATIONS          0
RECORDS                0
```

A10

```
F       F1,F2,F3,F4,F5,F6,F7.
V       P,Q,
        (F1(A,B,C)*F2(A,B,C)*F3(A,B,C)*F4(A,B,C))*(A**3*PDQ**2*P(MU))
K1     O F4(D1+,D2+,C)*F2(D1+,D2+,C)=F6(D1,D2)=
*       EQUALS

SYMBOLS             I=9, A, B, C.

INDICES             MU.

VECTORS             P, Q.

FUNCTIONS           D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3, UBG, DD, F1, F2, F3,
                    F4, F5, F6, F7.




* P(MU)*F1(A,B,C)*F2(A,B,C)*F3(A,B,C)*F4(A,B,C)

* ( A**3*PDQ**2 )


 +0.

RUNNING TIME            5
NUMBER OF TERMS         1
EQUAL TERMS             0
CANCELLATIONS           0
RECORDS                 0
```

A=

```
F       F1,F2,F3,F4,F5,F6,F7.
V       P,Q,
        (F1(A,B,C)*F2(A,B,C)*F3(A,B,C)*F4(A,B,C))*(A**3*PDQ**2*P(MU))
K1     0A**3* F2(D1+,D2+,C)*F4(D1+,D2+,C)=F6(D1,D2)=
*       EQUALS

SYMBOLS         I=9, A, B, C.

INDICES         MU,

VECTORS         P, Q.

FUNCTIONS       D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3, UBG, DD, F1, F2, F3,
                F4, F5, F6, F7,




+ P(MU)*F1(A,B,C)*F3(A,B,C)*F6(A,B)

* ( PDQ**2 )


 +0,

RUNNING TIME          5
NUMBER OF TERMS       1
EQUAL TERMS           0
CANCELLATIONS         0
RECORDS               0
```

```
F       F1,F2,F3,F4,F5,F6,F7.
V       P,Q,
        (F1(A,B,C)*F2(A,B,C)*F3(A,B,C)*F4(A,B,C))*(A**3*PDQ**2*P(MU))
K1     NA**D3** F2(D1+,D2+,C)*F4(D1+,D2+,C)=F6(D1,D2)*F7(D3)=
*       EQUALS

SYMBOLS          I=9, A, B, C.

INDICES          MU.

VECTORS          P, Q.

FUNCTIONS        D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3, UBG, DD, F1, F2, F3,
                 F4, F5, F6, F7.




+ P(MU)*F1(A,B,C)*F3(A,B,C)*F6(A,B)*F7(3)

* ( PDQ**2 )



 +0.

RUNNING TIME           4
NUMBER OF TERMS        1
EQUAL TERMS            0
CANCELLATIONS          0
RECORDS                0
```

```
F       F1,F2,F3,F4,F5,F6,F7.
V       P,Q,
        (F1(A,B,C)*F2(A,B,C)*F3(A,B,C)*F4(A,B,C))*(A**3*PDQ**2*P(MU))
K1     OP(D3+) * F2(D1+,D2+,C)*F4(D1+,D2+,C)=F6(D1,D2)*F7(D3)=
*       EQUALS

SYMBOLS        I=9, A, B, C.

INDICES        MU.

VECTORS        P, Q.

FUNCTIONS      D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3, UBG, DD, F1, F2, F3,
               F4, F5, F6, F7.




* F1(A,B,C)*F3(A,B,C)*F6(A,B)*F7(MU)

* ( A**3*PDQ**2 )


 +0.

RUNNING TIME           4
NUMBER OF TERMS        1
EQUAL TERMS            0
CANCELLATIONS          0
RECORDS                0
```

A14

```
F      F1,F2,F3,F4,F5,F6,F7.
V      P,Q,
       (F1(A,B,C)*F2(A,B,C)*F3(A,B,C)*F1(A,B,C))*(A**3-PDQ**2*P(MU))
L     F A=(B+C)*(B-C)=
M      F1+(D1+,B,D2+)=D1+D2**2=
*      EQUALS

SYMBOLS        I=9, A, B, C.

INDICES        MU.

VECTORS        P, Q.

FUNCTIONS      D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3  UBG, DD, F1, F2, F3,
               F4, F5, F6, F7.



+ P(MU)

* ( - B**8*C**6*PDQ**2 + 3.*B**10*C**4*PDQ**2 - 3.*B**12*C**2*PDQ**2 + B**14*PDQ**2 )


 +0.

RUNNING TIME        776
NUMBER OF TERMS   40004
EQUAL TERMS       29568
CANCELLATIONS     10428



RECORDS               0
```

A15

```
F       F1,F2,F3,F4,F5,F6,F7.
V       P,Q,
        (F1(A,B,C)*F2(A,B;C)*F3(B,A,C)*F4(A,B,C))*(A**3*PDQ**2*P(MU))
K       F2+(D1+,D2+,C)*F3+(D2+,D1+,C)=F7(D1,D2,F2,F3)=
*       EQUALS

SYMBOLS          I=9, A, B, C.

INDICES          MU.

VECTORS          P, Q.

FUNCTIONS        D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3, UBG, DD, F1, F2, F3,
                 F4, F5, F6, F7.
```

```
+ P(MU)*F1(A,B,C)*F7(A,B,F2,F3)*F4(A,B,C)

* ( A**3*PDQ**2 )


 +0.

RUNNING TIME           4
NUMBER OF TERMS        1
EQUAL TERMS            0
CANCELLATIONS          0
RECORDS                0
```

A16

```
C       K - PI + MU + NU. WITH AXIAL MAGNETIC TERM IN LEPTON PART.
C       NOTATIONS FOR MASS AND MOMENTUM -
C       K   -   MK, K.     .   K-CHARGE IS +.
C       PI   -   MP.
C       MU   -   M, Q.
C       NU   -   MN, QP.
C       W  =  POLARIZATION DIRECTION.
C
C
F       DIA=0.
S       LA=C,KSI=C,F1,M,MN,MK,MP.
V       Q,QP,K,W.
I       MU.NU,MUP,NUP,I1,I2.
B       LA,LAQ,KSI,KSIG,F1.
            (DIA(MU,NU)*CONJG*(DIA(MUP,NUP)))
J1      DIA(MU+,NU+)=UBG(I1,MN,QP)*(G(I1,MU)*G6(I1)+I*LA*(G(I1,MU)*G(I1,
        NU)-G(I1,NU)*G(I1,MU))*M**(-1)*(Q(NU)+QP(NU))*G5(I1))*0,5*
        (GI(I1)+I*G(I1,W)*G5(I1))  *UG(I1,-M,Q)*(F1*2.*K(MU)+(KSI-F1)*
        (Q(MU)+QP(MU)))=
J5      SPIN1,I1=
J7,4    TRIOK,TRACE,I1=
K5      MN=0=
K5      WDQ=0=
K5      WDQP=0=
K5      WDK=0=
K5      QPDQP=0=
K6      WDW=1=
K6      QDQ=-M**2=
K6      KDK=-MK**2=
*       YEP

SYMBOLS         I=2, LA=3, LAQ, KSI=3, KSIQ, F1, M, MN, MK. MP.

INDICES         MU. NU, MUP, NUP, I1, I2.

VECTORS         Q, QP, K, W.

FUNCTIONS       D, EPF=9. G=9, GI, G5=9, G6=3, G7, UG=3, UBG. DD, DIA=21.
```

A17

```
S       RLA,ILA,RKSI,IKSI,EPSC.
B       RLA,ILA,RKSI,IKSI,F1.
C
C
C       SUBSTITUTIONS LIKE THE FOLLOWING ONES USUALLY GREATLY INCREASE
C       THE NUMBER OF TERMS TO BE SORTED IN THE OUTPUT. THEREFORE
C       IT IS ADVISABLE TO FIRST DO SORTING, AND USING THE SORTED RESULT
C       AS NEW INPUT ON WHICH THESE SUBSTITUTIONS CAN THAN BE PERFORMED.
C       THIS IS ACHIEVED BY THE  YEP  COMMAND.
C
C
L1      QDQP=-0.5*MP**2+0.5*MK**2+QDK+QPDK+0.5*M**2=
C
C       RLA= REAL PART OF LA
C       ILA= IMAGINARY PART OF LA
C       RKSI= REAL PART OF KSI
C       IKSI= IMAGINARY PART OF KSI.
C
L5      LA=RLA+I*ILA=
L5      LAG=RLA-I*ILA=
L5      KSI=RKSI+I*IKSI=
L5      KSIG=RKSI-I*IKSI=
R


+ F1**2

* ( 6.*M**2*MK**2 + 2.*M**2*MP**2 - 4.*M**2*QDK + 12.*M**2*QPDK - 2.*M**4 - 8.*MK**2*MP**2

 + 16.*MK**2*QDK + 16.*MK**2*QPDK + 8.*MK**4 + 32.*QDK*QPDK )


+ RKSI**2

* ( - 2.*M**2*MK**2 + 2.*M**2*MP**2 - 4.*M**2*QDK - 4.*M**2*QPDK - 2.*M**4 )


+ IKSI**2

* ( - 2.*M**2*MK**2 + 2.*M**2*MP**2 - 4.*M**2*QDK - 4.*M**2*QPDK - 2.*M**4 )
```

A18

```
 + RLA*F1**2
 * ( - 16.*M**2*MK**2 - 8.*M**2*QDK - 8.*M**2*QPDK + 16.*MK**2*MP**2 - 40.*MK**2*QDK
  - 24.*MK**2*QPDK - 16.*MK**4 + 8.*MP**2*QDK - 8.*MP**2*QPDK - 16.*QDK**2 - 16.*QPDK**2
  - 32.*QDK*QPDK )


 + RLA**2*F1**2
 * ( - 16.*M**(-2)*MK**2*QDK**2 - 16.*M**(-2)*MK**2*QPDK**2 + 32.*M**(-2)*MK**2*QDK*QPDK
  + 16.*M**2*MK**2 + 16.*M**(-2)*MP**2*QDK**2 + 16.*M**(-2)*MP**2*QPDK**2 - 32.*M**(-2)*MP**2
 *QDK*QPDK - 32.*M**(-2)*QDK**3 - 32.*M**(-2)*QPDK**3 + 32.*M**(-2)*QDK*QPDK**2 + 32.*M**(-2)
 *QDK**2*QPDK - 16.*MK**2*MP**2 + 32.*MK**2*QDK + 32.*MK**2*QPDK + 16.*MK**4 - 16.*QDK**2
  + 48.*QPDK**2 + 32.*QDK*QPDK )


 + ILA**2*F1**2
 * ( - 16.*M**(-2)*MK**2*QDK**2 - 16.*M**(-2)*MK**2*QPDK**2 + 32.*M**(-2)*MK**2*QDK*QPDK
  + 16.*M**2*MK**2 + 16.*M**(-2)*MP**2*QDK**2 + 16.*M**(-2)*MP**2*QPDK**2 - 32.*M**(-2)*MP**2
 *QDK*QPDK - 32.*M**(-2)*QDK**3 - 32.*M**(-2)*QPDK**3 + 32.*M**(-2)*QDK*QPDK**2 + 32.*M**(-2)
 *QDK**2*QPDK - 16.*MK**2*MP**2 + 32.*MK**2*QDK + 32.*MK**2*QPDK + 16.*MK**4 - 16.*QDK**2
  + 48.*QPDK**2 + 32.*QDK*QPDK )


 + RKSI*F1
 * ( 4.*M**2*MK**2 - 4.*M**2*MP**2 + 8.*M**2*QDK - 8.*M**2*QPDK + 4.*M**4 )
```

A14

```
+ RLA*RKSI*F1
* ( 8.*M**2*QDK + 8.*M**2*QPDK + 8.*MK**2*QDK - 8.*MK**2*QPDK - 8.*MP**2*QDK + 8.*MP**2*QPDK
 + 16.*QDK**2 - 16.*QPDK**2 )


+ ILA*IKSI*F1
* ( 8.*M**2*QDK + 8.*M**2*QPDK + 8.*MK**2*QDK - 8.*MK**2*QPDK - 8.*MP**2*QDK + 8.*MP**2*QPDK
 + 16.*QDK**2 - 16.*QPDK**2 )


+ EPF(W,K,QP,Q)*ILA*F1**2
* ( 32.*I*M**(-1)*QDK - 32.*I*M**(-1)*QPDK + 16.*I*M )


+ EPF(W,K,QP,Q)*IKSI*F1
* ( - 16.*I*M )


+ EPF(W,K,QP,Q)*RLA*IKSI*F1
* ( 16.*I*M )


+ EPF(W,K,QP,Q)*ILA*RKSI*F1
* ( - 16.*I*M )


 +0.
```

A20

```
S       A1,A2,A3.
V       P1,P2,P3,Q1,Q2,Q3.
I       I1,I2,I3,J1,J2,J3.
F       F1,F2,F3,AF,BF,CF.
A       A1=3,A2=-4,P1=7,P2=-5,F1=3,F2=7,
        (A1**2+A1**-2+A1*A4+A1/A4+P1DP2+F1(A1,A2)*F2(A2)*A1*P1DQ1*P2(I1)
        +P1(4)**3+P1(4)+P1(4)**2+P1(4)**-1+P1(4)**-2)
J       ASYMP,-6=
*       EQUALS

SYMBOLS          I=9, A1, A2, A3, A4.

INDICES          I1, I2, I3, J1, J2, J3.

VECTORS          P1, P2, P3, Q1, Q2, Q3.

FUNCTIONS        D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3, UBG, DD, F1, F2, F3,
                 AF, BF, CF.
```

```
 + A1*A4 + A1*A4**(-1) + A1**2 + A1**(-2) + P1(4) + P1(4)**2 + P1(4)**3 + P1DP2



+ P2(I1)*F1(A1,A2)*F2(A2)

* ( A1*P1DQ1 )



 +0.

RUNNING TIME           9
NUMBER OF TERMS        9
EQUAL TERMS            0
CANCELLATIONS          0
RECORDS                0
```

A21

```
S       A1,A2,A3.
V       P1,P2,P3,Q1,Q2,Q3.
I       I1,I2,I3,J1,J2,J3.
F       F1,F2,F3,AF,BF,CF.
        (UBG(J1,A1,P1)*AF(2,I1,I2)*G(J1,J2)*G(J1,Q1)*BF(1,I3,I1)*
         UG(J1,A2,P2)*G(J1,Q2)*AF(2,I2,I3)*G(J3,Q3) )
J       ORDER,BF,2,I1,I2,I3=
*       EQUALS

SYMBOLS          I=9, A1, A2, A3.

INDICES          I1, I2, I3, J1, J2, J3.

VECTORS          P1, P2, P3, Q1, Q2, Q3.

FUNCTIONS        D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3, UBG, DC, F1, F2, F3,
                 AF, BF, CF.




+ BF(1,I3,I1)*AF(2,I1,I2)*AF(2,I2,I3)*UBG(J1,A1,P1)*G(J1,J2)*G(J1,Q1)*UG(J1,A2,P2)*G(J1,Q2)

*G(J3,Q3)




 +0.

RUNNING TIME            9
NUMBER OF TERMS         1
EQUAL TERMS             0
CANCELLATIONS           0
RECORDS                 1
```

```
S       A1,A2,A3.
V       P1,P2,P3,Q1,Q2,Q3.
I       I1,I2,I3,J1,J2,J3.
F       F1,F2,F3,AF,BF,CF.
        (UBG(J1,A1,F1)*AF(2,I1,I2)*G(J1,J2)*G(J1,Q1)*BF(1,I3,I1)*
         UG(J1,A2,P2)*G(J1,Q2)*AF(2,I2,I3)*G(J3,Q3) )
J       ORDEI,I1,I2,I3=
*       EQUALS

SYMBOLS          I=9, A1, A2, A3.

INDICES          I1, I2, I3, J1, J2, J3.

VECTORS          P1, P2, P3, Q1, Q2, Q3.

FUNCTIONS        D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3, UBG, DD, F1, F2, F3,
                 AF, BF, CF.
```

```
+ AF(2,I1,I2)*AF(2,I2,I3)*BF(1,I3,I1)*UBG(J1,A1,P1)*G(J1,J2)*G(J1,Q1)*UG(J1,A2,P2)*G(J1,Q2)

*G(J3,Q3)
```

```
 +0.

RUNNING TIME          9
NUMBER OF TERMS       1
EQUAL TERMS           0
CANCELLATIONS         0
RECORDS               0
```

A23

```
S       A1,A2,A3.
V       P1,P2,P3,Q1,Q2,Q3.
I       I1,I2,I3,J1,J2,J3.
F       F1,F2,F3,AF,BF,CF.
        (P1(I1)+P2(I1)+P3(I1))*(A1*P1(I1)+A2*P2(I1)+A3*P3(I1)+Q1(I1) )
J       ORTHO,Q1,P1,P2=
*       EQUALS

SYMBOLS         I=9, A1, A2, A3.

INDICES         I1, I2, I3, J1, J2, J3.

VECTORS         P1, P2, P3, Q1, Q2, Q3.

FUNCTIONS       D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3, UBG, DD, F1, F2, F3,
                AF, BF, CF.
```

```
+ A1*P1DP1 + A1*P1DP2 + A1*P1DP3 + A2*P1DP2 + A2*P2DP2 + A2*P2DP3 + A3*P1DP3 + A3*P2DP3

+ A3*P3DP3 + P3DQ1
```

```
 +0.

RUNNING TIME           8
NUMBER OF TERMS       10
EQUAL TERMS            0
CANCELLATIONS          0
RECORDS                0
```

```
S       A1,A2,A3.
V       P1,P2,P3,Q1,Q2,Q3.
I       I1,I2,I3,J1,J2,J3.
F       F1,F2,F3,AF,BF,CF.
        (P1(I1)+p2(I1)+P3(I1))*(A1*P1(I1)+A2*P2(I1)+A3*P3(I1)+Q1(I1) )
J       ORTHN,P1,P2,P3=
*       END

SYMBOLS           I=9, A1, A2, A3.

INDICES           I1, I2, I3, J1, J2, J3.

VECTORS           P1, P2, P3, Q1, Q2, Q3.

FUNCTIONS         D, EPF=9, G=9, GI, G5=9, G6=3, G7, UG=3, UBG, DD, F1, F2, F3,
                  AF, BF, CF.




 + A1  + A2  + A3  + P1DQ1  + P2DQ1  + P3DQ1



 +0.

RUNNING TIME            7
NUMBER OF TERMS         6
EQUAL TERMS             0
CANCELLATIONS           0
RECORDS                 0
```

A25