JOURNAL OF COMPUTATIONAL PHYSICS 8, 369–381 (1971)

Canonical Transformation by Computer (Foldy–Wouthuysen Transformation)

BERNARD ROSEN

Department of Physics, Stevens Institute of Technology, Hoboken, New Jersey 07030

Received January 22, 1970

A method for calculating canonical transformations by computer is illustrated by a specific example. Techniques, with wider application, for handling Dirac gamma matrix algebra, Heisenberg commutation relations and most of vector analysis were developed. The programs used the IBM FORMAC routines; the techniques developed here indicate applications of FORMAC to noncommutative algebras by means of list processing within the syntax of those routines.

The particular canonical transformation considered was that of Foldy and Wouthuysen on the Dirac Hamiltonian with electromagnetic potentials included. The calculations were carried out to the point at which all the so-called even operators up to the fourth order in the reciprocal of the mass had been generated.

I. INTRODUCTION

Canonical transformations provide a means of generating new conjugate variables (in the Hamiltonian sense) from old ones. In terms of these new variables, the transformed Hamiltonian of the system may be one with which calculations can be more readily done. One outstanding example of this is the transformation from the Schroedinger to the interaction picture in quantum electrodynamics.

While the algorithm for generating a given canonical transformation can be quite straightforward, the algebraic work can be formidable. This is especially true in quantum mechanical calculations wherein operators have algebraic properties in several different spaces (Hilbert space, spin space, space-time).

The purpose of this paper is to report on a method for carrying out a canonical transformation by computer using algebraic manipulation techniques. The programs discussed are based upon the FORMAC[1] (Formula Manipulation Compiler) routines developed by Sammet [1] and Tobey *et al.* [2] at IBM. These are a set of subroutines for the performance of the symbolic manipulation involved in ordinary algebra and elementary differential calculus. We report here on the use

[1] These are available as a super set of PL/I on System/360.

of these routines to perform analytically the steps of a Foldy–Wouthuysen [3, 4] FW transformation of the Dirac relativistic Hamiltonian with the inclusion of external electromagnetic fields. The even terms (defined below) of the transformed Hamiltonian up to those of fourth order in the reciprocal of the mass have been obtained.

The specific problem treated is actually rather complicated; the algebraic considerations include those connected with Dirac gamma matrices, the Heisenberg commutation relations, and, most importantly, three-dimensional vector analysis. The author has devised a list-processing technique with the syntax of FORMAC that permits noncommutative algebra to be carried out.

In the next section we shall present some background information on canonical transformations and on the Foldy–Wouthuysen transformation. In addition we shall outline the general procedure followed in our calculations. There are no new results in that section.

In Section III we will describe the particular list structure used in the programs and indicate those details of the programs which we believe are important. Finally we shall present the results.

II. Mathematical Background

To effect a quantum-mechanical canonical transformation on a Hamiltonian H, one must first find the appropriate generating function (an operator) [5]. Given this operator, say S, one has that the transformation is given by

$$H' = e^S H e^{-S} - ie^S(\partial e^{-S}/\partial t), \tag{1}$$

where H' is the transformed Hamiltonian. In practice, this may have to be carried out to some order by a truncated version of the formules

$$e^S H e^{-S} = \sum_{n=0}^{\infty} \frac{1}{n!} C^{(n)}(S, H), \tag{2}$$

$$e^S \frac{\partial e^{-S}}{\partial t} = \sum_{n=0}^{\infty} \frac{1}{(n+1)!} C^{(n)}\left(S, \frac{\partial S}{\partial t}\right). \tag{3}$$

In these formulas the quantities $C^{(n)}$ represent the n-th repeated commutator of S with the other operator indicated.

In addition to the complexities introduced by the performance of the commutations implied by the previous two formulas, there is the difficulty that S itself may

be unknown and have to be developed in a series. More formally, assume that one is given a Hamiltonian of the form

$$H = H_0 + \lambda H_1 , \tag{4}$$

wherein H_0 is a piece with which calculations may be done readily and in which the piece λH_1 has certain undesirable properties. The quantity λ is a scalar parameter which can be presumed small. A canonical transformation which will eliminate the term λH_1 is sought; however, the operator S is unknown. If one can solve the equation

$$[H_0 , S_1] = H_0 S_1 - S_1 H_0 = \lambda H_1 \tag{5}$$

for S_1, then one can use the operator S_1 to eliminate λH_1 from H by means of (2) and (3). This will usually be at the expense of adding terms that are at least quadratic in λ to H.

The original problem of finding one canonical transformation to eliminate the term λH_1 can now be changed to the following: Find a succession of canonical transformations, generated in turn by

$$S_1 , S_2 , ..., S_n ,$$

where the commutator of H_0 with S_i is equal to the objectionable terms remaining in the (transformed) Hamiltonian after the previous transformations have been carried out. Since the processes indicated each generate an infinite series of terms, one clearly must truncate all expressions at some power of λ to which one intends the calculation to be consistent.

The entire process is straightforward if one can generate the S's. It can easily become tedious; the possibility that the entire process can be computerized suggests itself. The remainder of this paper concerns itself with a specific application of this class of algebraic operations.

The particular case chosen was that of the FW transform of the Dirac Hamiltonian

$$H = m \left(\beta + \frac{\boldsymbol{\alpha}}{m} \cdot (\mathbf{p} - q\mathbf{A}) + \frac{q\phi}{m} \right) \tag{6}$$

for a relativistic electron. The quantities and operators in this Hamiltonian have the following significance. The mass is denoted by m while A and ϕ are the electromagnetic vector and scalar potentials, respectively. The charge on the electron is q. The Hamiltonian is understood to act on a column vector of dimension four standing to the right whose components are functions of the space-time coordinates. This vector is called a spinor and the four matrices α_x, α_y, α_z and β are 4×4

matrices acting on it. Finally, $\mathbf{p}$ stands for the momentum operator of single-particle quantum mechanics so that $\mathbf{p} \equiv -i\nabla$. In all these formulas the so-called natural units ($\hbar = 1$, $c = 1$) have been used.

While the computational features of this problem are of prime interest, here let me state the physical motivation for the FW transformation. This lies in the fact that a single-particle interpretation of the states connected with the Dirac Hamiltonian appears to be physically impossible. In the nonrelativistic limit one should be able to recover the Pauli-Schroedinger theory of the spinning electron. By this nonrelativistic limit we mean that not only should the kinetic energy of the particle be small compared to the rest mass energy by also that the frequencies and wave-numbers characterizing the external fields should be small compared to m.

As just indicated, the Pauli–Schroedinger theory should be the limit of the Dirac theory if the proper expansion in the reciprocal of the mass were to be carried out. The key point shown by Foldy and Wouthuysen and by Tani is that the correct expansion is obtained by a canonical transformation which eliminates the so-called odd operators from the Hamiltonian. In essence, the odd operators contain those *matrices* which couple the spinor components of the negative energy eigenfunction in (6) with those of positive energy in the rest frame of the particle. Their elimination permits the separation of the two types of eigenfunctions.

In the usual representation the odd operator matrices (in block diagonal form) are

$$\alpha = \begin{pmatrix} 0 & \underline{\sigma} \\ \underline{\sigma} & 0 \end{pmatrix}, \qquad \Upsilon = \begin{pmatrix} 0 & \underline{\underline{\sigma}} \\ -\underline{\underline{\sigma}} & 0 \end{pmatrix}, \qquad \gamma_5 = \begin{pmatrix} 0 & \underline{1} \\ \underline{1} & 0 \end{pmatrix}, \qquad \gamma_5\beta, \tag{7}$$

while the others, the even ones, are

$$\underline{\underline{1}} = \begin{pmatrix} 0 & \underline{1} \\ \underline{1} & 0 \end{pmatrix}, \qquad \beta = \begin{pmatrix} \underline{1} & 0 \\ 0 & -\underline{1} \end{pmatrix}, \qquad \underline{\underline{\sigma}} = \begin{pmatrix} \underline{\sigma} & 0 \\ 0 & \underline{\sigma} \end{pmatrix}, \qquad \beta\underline{\underline{\sigma}}. \tag{8}$$

The σ_i are 2×2 Pauli spin matrices, $i = k, 2, 3$, and $\underline{1}$ is the twodimensional unit matrix. In this representation the odd operators are those which couple the upper two components of the spinors with the lower two components.

As has been stated, the FW transformation is effected by the elimination of the odd operators from the Dirac Hamiltonian. If the four vector (A, ϕ) is identically zero, then the use of

$$S = \left[\frac{\beta\alpha \cdot p}{2\,|\,p\,|} \arctan\left(\frac{|\,p\,|}{m}\right)\right] \tag{9}$$

in the formulas given above will give

$$H' = \beta(p^2 + m^2)^{1/2}.$$

In the general case the operator S cannot be determined, so far as is known, in closed form. Instead, a succession of S's, each of a higher power in the reciprocal of the mass can be determined and successive transformations be carried out. Suppose one writes the Hamiltonian at any stage of the calculations in the form $H = \beta m + \underline{E} + \underline{O}$, where $\underline{O}$ is the part containing odd operators only and $\underline{E}$ contains even operators only then the choice for the S to use at this stage is

$$S = \beta \underline{O}/2m. \tag{10}$$

Then H' calculated according to (1) is at least one order lower in mass than is H since $\beta \underline{O} + \underline{O}\beta = 0$ for all odd operators.

The method employed for systematically carrying out the FW transformation is as follows:

(a) The operator S is formed according to (5).

(b) The canonical transformation with this S is carried out to a given order, say N, in the reciprocal of the mass using (1) in the form given by the formulas (2) and (3).

(c) Given the transformed Hamiltonian that contains odd operators multiplied by powers of the reciprocal mass at least as high as some order r, one can repeat the steps to obtain a Hamiltonian which is of order $r + 1$ in the reciprocal mass[2].

As mentioned above, the calculation on which we report was carried out to the fourth power in the inverse of the mass. At this point all parts of the various programs had been tested and the algorithms appeared to be correct. As indicated below, further computations on our particular computer, a 360/40 Model H, are not practicable.

III. Implemenation

A. *General Considerations*

In order that the reader will be better able to follow the discussion, we shall first give a particular example of a commutator that has to be calculated, viz.,

$$[\boldsymbol{\sigma} \cdot \mathbf{B}, \boldsymbol{\alpha} \cdot \mathbf{p}] = \boldsymbol{\sigma} \cdot \mathbf{B}\boldsymbol{\alpha} \cdot \mathbf{p} - \boldsymbol{\alpha} \cdot \mathbf{p}\boldsymbol{\sigma} \cdot \mathbf{B}. \tag{11}$$

If we recall that p stands for $(-i)$ times the gradient operator, then we see that two separate algebraic structures are reflected in the commutator. One is the Dirac-

[2] At some point one can see that no more even operators of small enough order will be generated by succeeding transformations. Thus not all steps need be done.

Pauli algebra of the matrices σ and α and the other is the Heisenberg algebra of the well-known commutation relations of quantum mechanics. It is necessary, therefore, to use the full algebra of the Dirac matrices and not just their commutation relations. One has to calculate, in fact,

$$\gamma^5[B_i, p_i] + i\alpha_k\{B_i, p_j\}\,\epsilon_{ijk} \equiv \gamma^5(\mathbf{B}\cdot\mathbf{p} - \mathbf{p}\cdot\mathbf{B}) + i\boldsymbol{\alpha}\cdot(\mathbf{B}\times\mathbf{p} - \mathbf{p}\times\mathbf{B}). \quad (12)$$

As the reader can discern, the curly brackets stand for the anticommutator and the ϵ_{ijk} is the Levi–Civitta density. It should be remembered that the Hamiltonian of which these terms will form a part acts on a spinor wave function whose presence on the right is understood.

The only method to use for a machine calculation of the commutator or anticommutator that occurred to the author is that which imitates an ordinary hand calculation, viz., bring all differentiation operators to the right factor by factor within each term. A straightforward algorithm using list-processing was developed for this purpose and will be described below.

A glance at the previous Eq. (2) reveals that feature of the programming which the author found most formidable. This is the treatment of ordinary vector analysis. The goal of this calculation was not to obtain pages of complicated formulas but rather to see if any pattern might emerge in the higher order terms. It was hoped to completely avoid vector and tensor indices with their attendant proliferation of Kronecker deltas and Levi–Civitta densities. However, the indicated method for carrying out the Heisenberg algebra caused a gradient operator to be separated physically (i.e., positionally on the list) from any vector product. With proximity destroyed as an indicator of these possible multiplicative relationships between vectors, reliance on vector and tensor indices has to be made. Some effort, described below, was then made to achieve algebraic simplification by restoring ordinary notation wherever possible. Let me say, though, that Kronecker deltas were completely avoided and that no Levi–Civitta densities with common indices occurred in any completely processed term.

As has been stated, the algebra of the dynamical commutation relations was performed using a list-processing technique. This was done within the framework of FORMAC using the syntactic entity *unspecified function* [1, 2] permitted by that language. As in the ordinary mathematical notation, when we write "Let z be a function f of x and y; $z = f(x, y)$" without any further specificiation of f, so one can have the FORMAC statement

$$\text{LET}[Z = F\cdot(X, Y)]$$

with very much the same meaning. From our point of view, the unspecified function acts both as a *labelled list of factors whose order is not subject to change by the automatically invoked simplification routines of FORMAC* and also as a factor within a given term in the ordinary algebraic sense.

Every term that occurs, whether in the operator S, in the Hamiltonian H, or in $\partial S/\partial t$, in these programs has the structure[3]

(algebraic constant) * (name of Dirac matrix) * (unspecified function).

The algebraic constant can be an implicit unity. The names of Dirac matrices were GAMMA, BETA, ALPHA, UNITY, $G5$ (for γ 5), $S5$(for $\beta\gamma$ 5), SIGMA, and AXVECT (for $\beta\sigma$). Four of these (BETA, UNITY, $S5$, $G5$) are rotational scalars; the other four are each three-dimensional vectors. This means that for each name given there are actually three distinct matrices whose distinguishing subscripts bear the significance of vector indices in ordinary three-dimensional space.

If the Dirac matrix in a term, and there is only one per term, is a scalar then the name of the unspecified function is NONE; otherwise *the vector index of the matrix is taken for the name.* A group of examples, with some additional comments, will now be given. In each case, we give the term using ordinary vector notation, followed by the equivalent using vector indices and the Einstein summation convention and, finally, we give the FORMAC representation we used:

$$\boldsymbol{\alpha} \cdot \mathbf{E} = \alpha_i E_i .$$

ALPHA * LABEL (1). (EFIELD, LABEL. (1))

$$i\boldsymbol{\sigma} \cdot \nabla\phi \equiv i\sigma_j \nabla_j \phi.$$

#1 * SIGMA * LABEL (1). (DEL, LABEL. (1), PHI, NONE)

$$\boldsymbol{\gamma} \cdot (\mathbf{B} \times \nabla) = \gamma_i B_j \nabla_k \epsilon_{ijk} .$$

GAMMA * LABEL (1). (BFIELD, LABEL. (2), DEL, LABEL. (3)) * EPSILON. (LABEL. (1), LABEL. (2), LABEL. (3))

$$(1/m)\, \beta\, \nabla \cdot E.$$

(1/MU) * BETA * NONE, (DEL, LABEL. (1), EFIELD, LABEL. (1)) or
(1/MU) * BETA * NONE. (DIV. (EFIELD), NONE)

First we note that any term in the Hamiltonian is a rotational scalar; therefore, every vector (or tensor) index occurs twice and the summation convention holds. Every argument list for an unspecified function used in this manner consists of dyads each in turn made up of a quantity and its index. These conventions insure a uniform structure for the lists, which in turn simplifies the processing. This structure has proved sufficiently flexible to have been used in all steps of the programs.

[3] There are also a possible ϵ_{ijk}'s.

Note that within the list itself the indices are also unspecified functions. This feature facilitates obtaining the number of the index which under FORMAC rules is the second argument of the unspecified LABEL. The name, LABEL, is the first argument. Furthermore it is necessary that the index may itself be a list; this occurs whenever the first element of a triad is tensorial in nature. As an example, the term

$$\sigma_j(\nabla_j E_i)\,\nabla_i$$

would have the representation

SIGMA * LABEL (1). (DEL. (EFIELD), INDICE. (LABEL. (1), LABEL. (2)),
DEL, LABEL.(2))

There is a fine point implicit in all this that may cause the reader some confusion. The name of the list, LABEL (1), is merely an indexed quantity which can serve as the name of an unspecified function. Although LABEL. (1) and LABEL(1) are quite different syntactically, they are semantically equivalent in our usage of them.

As a further example of the representation of terms we give the initial forms of S and of H. These are

H = BETA * MU * NONE. (1, NONE) + ALPHA * LABEL (1). (PI. LABEL.(1))
+ UNITY * NONE. (PHI, NONE),

S = (1/2/MU) * GAMMA * INDEX (1). (PI, INDEX.(1)),

in which we have used the name PI to stand for the mechanical momentum

$$\boldsymbol{\Pi} \equiv \mathbf{p} - \mathbf{qA}.$$

Note further that we consistently use the name INDEX in the terms of S and LABEL in the terms of H; this permits the programs to sort out the two sets of dummy indices without confusion.

Given this method for forming lists, certain aspects of list processing are quite easy within the framework of FORMAC. The macro NARGS applied to an unspecified function yields the total number of arguments of the function plus one for the function name. The function ARG can be used to isolate any element on the list directly. This is actually easier than in LISP. If the list (unspecified function) is a factor within a term it can be isolated by means of a FORMAC expression such as

LET (LIST = TERM/EVAL (TERM, NAME. ($ (1)), 1));

where NAME is the unspecified-function name. Lists can be readily concatenated if the names of the unspecified functions can first be removed. This brings up the only basic inconvenience in regard to list processing with FORMAC; there is no

primitive function that yields the tail of a list. The author found that converting the algebraic formulas to PL/I character strings via the use of the CHAREX macro and then reintroducing the appropriate substring back into the FORMAC interpreter was the most convenient method of handling this difficulty.[4]

In many instances the programs converted the tensor symbolism into the more compact notation of vector analysis; such terms as DOT, CROSS, CURL, and DIV were used in contexts matching their ordinary use. For example,

ALPHA*LABEL(1). (EFIELD, LABEL.(2), BFIELD, LABEL.(2), PI, LABEL.(1)

was changed into

ALPHA * LABEL (1) · (DOT. (BFIELD, EFIELD), NONE, PI, LABEL.(1))

near the end of the program step called HSNBERG.[5] Note that lexicographraphical reordering has also taken place.

The FORMAC interpreter will recognize as distinct quantities some that are syntactically different but semantically the same. This can occur in this type of calculation because of the manner in which the names of dummy indices are assigned or because of the order in which functions appear in one of the argument lists mentioned above. This problem in turn was connected with that of preventing a proliferation of ϵ_{ijk}'s as the calculations progressed. A considerable amount of relabelling and lexicographical reordering was necessary to bring about the cancellation of grouping of terms that differed only by a numerical factor. By and large, this was done successfully and automatically by the programs. Advantage was taken of the fact that the results from the program step DIRAC, described below, occurred in pairs or quadrupules of terms.

B. *Details of Individual Programs*

1) *DIRAC.* The commutation of the Dirac matrices was completely straightforward and was carried out by what in effect was a table look up. Preliminary to this, the operator S is formed from the odd parts of H and also, the numerical factors in the several terms of H and S are isolated. The output was in pairs or quadruples of terms each of the form

(numerical coefficient) * (Dirac Matrix) *
OPERATOR. (LABEL (1). (...), MO, INDEX (1). (...)), where

MO stands for multiplicative operator.

[4] Alternatively, to go from $F \cdot (A, B, \ldots, P)$ to $F \cdot (B, C, \ldots, P)$ one can construct a list $F \cdot [\$(2), \$(3), \ldots]$ and use the EVAL function. This would be particularly effective if lists of definite length were being used.

[5] This represents another use of the unspecified function, viz, to indicate a specific operation, either binary or unary.

The term paired with that above has the factors leading off with LABEL and INDEX interchanged and also a possible difference in sign. The numerical coefficient may contain Levi–Civitta densities. The MO is one of the three (DOT, CROSS ON) and stand for the scalar, vector, and ordinary multiplicative operations of three-dimensional vector analysis, respectively. The MO 'ON' also occurs when the gradient operator is applied to a scalar function.

2) *HSNBRG.* The commutation of the momentum operators, the p_j with the external fields and potentials follows Heisenberg's famous formulas:

$$[p_j, f(\mathbf{r})] = -i\hbar\, \partial_j f, \tag{13}$$

$$[p_j, p_i] = 0. \tag{14}$$

Experience with the programs indicated that direct use of (1) and (2) led to a very large number of terms because of the expansion of powers of $(\mathbf{p} - q\mathbf{A})$. A further disadvantage of this form was that the gauge-dependent quantity div A made its appearance. For these reasons an alternative representation in terms of $\mathbf{\Pi} \equiv \mathbf{p} - q\mathbf{A}$ was employed. Equation (2) is now replaced by

$$\frac{1}{q}[\Pi_i, \Pi_j] = -i(\partial_i A_j - \partial_j A_i) = -i\epsilon_{ijk}B_k\,. \tag{15}$$

In order to carry out this type of commutation rule it was necessary to temporarily relabel the Π operators coming from S to say π and to give specific rules for moving a π to the right of a Π. Π's were not moved to the right of π. This artificial distinction between two essentially identical objects can lead to complicated expressions equivalent to zero; by relabelling indices and by lexicographical reordering of lists this was avoided.

As each operator is moved to the right, two new lists are generated; these new lists are subject to further list processing and one or both may eventually evaluate to zero. The method is recursive but it was not programmed using recursive subroutines. Instead, the lists were simply generated and marked for later exclusion or inclusion in the final answer.

3) *ADDUP.* The individual terms from HSNBRG were divided by the order of commutation (M) and then were added together by this simple routine. The results, constituting the M-th commutator of S with H, divided by M factorial were then added to previous results to form the transformed Hamiltonian to a given order.

This comparatively simple process of addition proved to be the most time-comsuming portion of the calculation since the number of operations required

should go up as the square of the number of terms in a sum. This is so because the term to be added must be compared with all terms in a sum to see if collection or cancellation of terms should occur. The use of subsums, each the coefficient of one type of Dirac matrix and each containing the same power of the mass, proved to be a useful technique. For example, it took almost an hour to add up some two hundred terms. This indicates that computations to higher order were not practicable for us.

4) *SDOT.* The differentiation of the operator S is carried out factor by factor by list processing because of the noncommutative properties of the factors. The quantities $\dot{A}$ and $\dot{B}$ are replaced by $(-E\text{-grad } \phi)$ and -curl E, respectively. This step was followed by a simplified version of HSNBRG.

5) *NEW-S.* The odd portions of H that will be used to select a new operator are easily picked out by a *PL/I* routine. The actual operator S is formed term by term in DIRAC.

These calculations were carried out on an IBM 360/40 Model *H*. This is the minimum size IBM machine on which one can use the FORMAC-*PL/I* interpreter.

IV. Results

We present here the transformed Hamiltonian (even terms) that results from the elimination of all odd operators whose order in η is greater than or equal to four. In the formulas given below we use the following notation:

$$\hbar = c = 1,$$

$$m = \text{mass},$$

$$Q = \text{charge}$$

$$\boldsymbol{\Pi} = \mathbf{p} - Q\mathbf{A} \equiv -i\nabla - Q\mathbf{A},$$

$$\boldsymbol{\pi} = \boldsymbol{\Pi}/m,$$

$$\mathbf{b} = Q\mathbf{B}/m^2,$$

$$\mathbf{e} = Q\mathbf{E}/m^2,$$

$$\partial = \nabla/m.$$

Further, while the operator π is considered to operate on the *wave function only* irrespective of its position relative to other operators; ∂ does not, on the other hand, act on the wave function.

The transformed Hamiltonian is given by

$$H' = \beta m\left(1 + \frac{\pi^2}{2} - \frac{\pi^4}{8} + \frac{(e^2 - b^2)}{8}\right) + Q\phi + m\left(-\frac{\partial \cdot \mathbf{e}}{8} - \frac{11\partial^2\partial \cdot \mathbf{e}}{128}\right.$$
$$- \frac{i(\partial^2\mathbf{e}) \cdot \pi}{12} - \frac{i(\pi \cdot \partial)(\partial \cdot \mathbf{e})}{6} + \frac{3(\partial \cdot \mathbf{e})\,\pi^2}{32} + \frac{i\mathbf{b} \cdot (\partial x\mathbf{e})}{12} + \frac{(e x \mathbf{b}) \cdot \pi}{32}$$
$$\left. + \frac{i(\mathbf{e} \cdot \partial x\mathbf{b})}{64} - \frac{i[\partial x(\partial x\mathbf{e})] \cdot \pi}{192} + \frac{5[(\partial \mathbf{e}) \cdot \pi] \cdot \pi}{32}\right)$$
$$- \frac{m\beta\sigma \cdot}{2}\left(\mathbf{b} + \frac{i(\pi \cdot \partial)\,\mathbf{b}}{2} + \frac{\partial^2\mathbf{b}}{4} - \frac{\mathbf{b}\pi^2}{2}\right) - \frac{m\sigma \cdot}{4}\left((e x \pi) + \frac{i(\partial x\mathbf{e})}{2}\right)$$
$$+ \frac{mi}{6}\,\partial(\sigma \cdot (e x \pi)) \cdot \pi + \frac{m\sigma \cdot}{6}\left(\frac{\mathbf{e} x(\partial x\mathbf{b})}{} - \frac{i(\partial x\mathbf{e})\,\pi^2}{2} - (e x \pi)\,\pi^2\right.$$
$$\left. + \frac{3}{8}\left(\mathbf{e} \times \frac{\partial \mathbf{e}}{\partial \tau}\right) + i(b \cdot e)\,\pi - \mathbf{b}i(e \cdot \pi) + \frac{i\partial^2(\partial x\mathbf{e})}{2} + \frac{(\partial^2\mathbf{e})\,x\pi}{2}\right.$$
$$\left. - (\pi \cdot \partial)(\partial x\mathbf{e}) + \frac{(\partial \mathbf{e}) \cdot \mathbf{b}}{2} + \frac{(\mathbf{b} \cdot \partial)\,\mathbf{e}}{2} - \frac{\mathbf{b}(\partial \cdot \mathbf{e})}{2}\right).$$

While the author exercised normal care and diligence in verifying the result just stated, let it be said that there are some possible sources of error not directly connected with the programs described above. Besides possible transcription errors, there are those connected with the necessity of using peripheral storage devices in these computations; a change of release in the Operating System for System 360 played considerable havoc with the input and output of data sets. Finally, in preparing the form of the result given above the author performed some further editing and collection of terms.

V. Summary

We have presented a method for performing noncommutative algebra within the frame work of an existing algebraic manipulation language PL/I-FORMAC. The application was to canonical transformations and, in particular, to the Foldy–Wouthuysen transformation of the Dirac Hamiltonian.

The method appears to us as one of general applicability to canonical transformations and to the calculation of commutators; only the specific rules for interchanging operators need to be changed in order that a different case can be treated.

ACKNOWLEDGMENTS

The author is indebted to the many members of the staff of the Computer Center at Stevens for information and help. In particular, he wishes to thank its former director, Dr. Irving Rabinowitz, for his encouragement of this work.

Stevens Institute was more than generous in providing time on the computer for this work and the author is very grateful.

REFERENCES

1. J. F. SAMMET, Formula manipulation by computer, *in* "Advances in Computers," (F. Alt and M. Rubinoff, Eds.) Vol. 8, Academic Press, New York, 1967.
2. R. TOBEY, "PL/I-FORMAC Interpreter User's Reference Manual," SHARE Contributed Program Library (Share 360D-03.3.004),
3. L. L. FOLDY AND S. A. WOUTHUYSEN, *Phys. Rev.* **78** (1950), 29.
4. S. TANI, *Progr. Theoret. Phys. Japan* **6** (1951), 267.
5. J. BJORKEN AND S. DRELL, "Relativistic Quantum Mechanics," Chapt. 4, McGraw-Hill, New York, 1967.