General Relativity and Gravitation, Vol. 25, No. 2, 1993

The Program ORTOCARTAN for Algebraic Calculations in Relativity

Andrzej Krasiński[1]

Received April 22, 1992

The program ORTOCARTAN can calculate the curvature tensors (Riemann, Ricci, Einstein and Weyl) from a given orthonormal tetrad representation of the metric tensor. It was first announced in 1981, but since then has undergone several extensions and transplants onto other computers. This article reviews the current status of the program from the point of view of a user. The following topics are discussed: the problems that the program can be applied to, the special features of the algorithms that make the program powerful, the technical requirements to run the program and two simple examples of applications.

1. WHAT CAN THE PROGRAM DO?

ORTOCARTAN takes the components of an orthonormal tetrad of exterior differential forms representing a metric tensor as the input data. Let $e^i = e^i{}_\alpha dx^\alpha$ be the forms ($i, j, k, \ldots$ label the forms and the tetrad components of tensors, $\alpha, \beta, \gamma, \ldots$ label the coordinate components), and let $[\eta_{ij}] = \mathrm{diag}(+1, -1, -1, -1)$. Any metric form can be represented either by

$$ds^2 = g_{\alpha\beta} dx^\alpha dx^\beta \tag{1}$$

or by

$$ds^2 = \eta_{ij} e^i e^j , \tag{2}$$

[1] N. Copernicus Astronomical Center, Polish Academy of Sciences, Bartycka 18, 00 716 Warszawa, Poland

0001-7701/93/0200-0165$07.00/0

where $g_{\alpha\beta} = \eta_{ij} e^i{}_\alpha e^j{}_\beta$. The input data for ORTOCARTAN are the components of the matrix $[e^i{}_\alpha]$.

The program calculates $\det[e^i{}_\alpha]$, the inverse matrix $[e^\beta{}_j]$ that is defined by

$$dx^\beta = e^\beta{}_j e^j, \tag{3}$$

the Ricci rotation coefficients $\Gamma^i{}_{jk}$ defined by

$$de^i = \Gamma^i{}_j \wedge e^j, \tag{4}$$

where $\Gamma^i{}_j = \Gamma^i{}_{jk} e^k$, the tetrad components of the Riemann tensor defined by

$$d\Gamma^i{}_j + \Gamma^i{}_s \wedge \Gamma^s{}_j = \tfrac{1}{2} R^i{}_{jkl} e^k \wedge e^l, \tag{5}$$

and the tetrad components of the Ricci and Weyl tensors. The main application of the program is for writing out the Einstein equations given the metric tensor and for verifying the solutions. In addition to the quantities listed, the program can also calculate the metric tensor, the inverse metric, the Christoffel symbols, the Einstein tensor and the coordinate components of the Riemann, Ricci, Einstein and Weyl tensors, with arbitrary positions (up or down) of each index. The main program is distributed together with the program CALCULATE which is a handy algebraic "abacus". CALCULATE was meant to allow the users of ORTOCARTAN to perform such algebraic operations as differentiation of symbolic expressions, simplifying the results of algebraic operations on complicated expressions, transforming symbolic expressions by series of substitutions, verifying solutions to differential equations, or any combination thereof. With the exception of calculating integrals and simplifying rational functions, the program should be able to perform all kinds of algebraic operations (see example in Table III, Section 4).

In contrast to several other systems, such as SHEEP [1,2] (now probably the main player in the applications of algebraic computing to relativity), in ORTOCARTAN the main emphasis was put not on the number of secondary programs for special tasks (such as the Petrov classification, the Segre classification of the Einstein tensor or calculating the Newman–Penrose spin coefficients), but on the power and generality of the algorithms. We[2] wanted to make sure that the program will not fail the user even with most complicated expressions to process, and with most sophisticated substitutions to perform in them. To what extent we succeeded can only be judged by the users. The next section is meant to encourage the candidates for users to give ORTOCARTAN a chance and try it.

[2] See Appendix A to find out who "we" are.

2. THE ADVANTAGES OF ORTOCARTAN'S ALGORITHMS

This section was partly inspired by reports written by users of other algebraic systems, published in conference proceedings and such journals as the *SIGSAM Bulletin* and the *Journal of Symbolic Computation*.[3] It seems that in addition to those programming problems that we wanted to solve, we managed to bypass other problems without knowing that they exist and cause trouble in other systems. In most of the cases, we succeeded by exploiting the most powerful device in any LISP system: recursive definitions.

ORTOCARTAN can handle rational powers of rational numbers. It will first simplify each rational number by detecting common factors of the numerator and the denominator, producing an integer whenever possible. Then, it will transform the number to such a form that the exponent is positive (by using the identity $(k/l)^{-a} = (l/k)^a$), the base is integer (by using the identity $(k/l)^{m/n} = l^{-m}(kl^{n-1})^{m/n}$), and the exponent is of the form 1/(integer) (by using $k^{m/n} = (k^m)^{1/n}$). Finally, it will try to make the denominator of the exponent as small as possible by using the identity $(k^{p\cdot q})^{1/(p\cdot l)} = (k^q)^{1/l}$. It will miss simplifications resulting from multiplying different bases for the same exponent, such as $\sqrt{6}\cdot\sqrt{2} = 2\sqrt{3}$, but such failures can be easily corrected through user-defined substitutions (see below).[4]

ORTOCARTAN has no problems making substitutions for sub-sums and sub-products in sums and products, respectively. For example, if the user defines the substitution $B+D = U$, then ORTOCARTAN will readily recognize that $(A+B+C+D+E)$ is to be substituted by $(A+C+U+E)$. Technically, such a substitution is equivalent to recognizing a given subset in a given set and replacing it with another subset. This is simple in ORTOCARTAN only because sums and products are automatically ordered in a unique way. To explain the principle of ordering would require going into too much detail of LISP for the taste of a typical reader of this journal; it is explained in the program's documentation [7]. The substitution for a part of a sum or a product can be missed only if the expression to be substituted for is multiplied by a factor or disguised in a more tricky way. The $B + D = U$ would be missed, for example, in $(B + 2D + E)$.

---

[3] Detailed references withheld in order to avoid making them seem negative.

[4] This kind of failure may occur only with numbers. In order to avoid it, each exponentiated integer would have to be factorized into prime factors, and the identity $(a\cdot b)^p(a\cdot c)^q = a^{p+q}\cdot b^p\cdot c^q$ would have to be applied. It was our feeling that, more often than in a simplification, this would result in making the representation of the number longer.

The arguments of functions can be functions themselves, or can be symbols for explicit algebraic expressions. The chain rule for differentiation will go into the arguments to an arbitrary depth, until it finds the independent variables or until the derivative is found to be zero. The user can use only the names of the functions in the formulae, or can write out explicitly the whole functional expressions (together with their arguments). The program will automatically pick up the style chosen by the user and continue the same style throughout the calculation.

While performing user-defined substitutions ORTOCARTAN automatically calls the algebraic simplification on the result of each single substitution. In some other systems, all the substitutions are first performed as simple replacements, and the algebraic simplification is called only afterwards. This can result in difficulties. The scheme followed by ORTOCARTAN guarantees that when a new substitution is added to a series of previous substitutions, then the new substitution will be performed exactly in the same expression that the user has seen previously.

The feature which is rarely reported in the literature on other systems is substitutions by pattern-matching. They can be most simply explained by examples. Suppose r is the symbol for $(x^2 + y^2 + z^2)^{1/2}$, where x, y and z are coordinates. The most economical way of doing a calculation with r is to define r as a function of (x, y, z) and then tell the program to substitute $r_{,x}$ by (x/r), $r_{,y}$ by (y/r) and $r_{,z}$ by (z/r). However, the 3 substitutions would normally all have to be written out explicitly. With pattern-matching, one can define some variables as members of a special class called MARKERS; suppose M is one of them. Then M will become a represent-anything symbolic variable, and the 3 substitutions can be defined by the single equation $r_{,M} = M/r$.

The usefulness of pattern-matching can be demonstrated by the following other examples:

(i) Suppose E is a small parameter, and the calculation is to be performed only up to terms linear in E. Instead of writing out all the substitutions $E^2 = 0$, $E^3 = 0$, ..., $E^n = 0$, up to the highest power that could possibly show up, it is enough to write $E^M = 0$.[5]

(ii) Suppose a can only be equal to $+1$ or -1. Instead of writing out the substitutions $a^2 = 1$, $a^3 = a$, $a^4 = 1$, and so on, one can write here

[5] ORTOCARTAN cannot automatically develop a functional expression into a truncated power series, but all the user has to do in order to perform an approximate calculation is to substitute the truncated series for the components of the metric tensor, define the appropriate expression for the determinant, and define the powers of the small parameter that should be neglected. Then all the expressions processed will become just polynomials in the small parameter.

$a^M = a^{(\text{remainder } M2)}$, and ORTOCARTAN will understand.
(iii) Suppose p is the symbol for $(t^2 - r^2)^{1/2}$, and you want to develop all integer powers of p so as to get rid of all fractional powers of $(t^2 - r^2)$ except possibly for $(t^2 - r^2)^{1/2}$. Then the appropriate substitution is:

$$p^M = (t^2 - r^2)^{(\text{quotient } M2)}(t^2 - r^2)^{(\text{remainder } M2)}.$$

The first factor in the above formula will, for each positive value of M, be an integer power of $(t^2 - r^2)$ and will be developed into a polynomial, while the second factor will be either 1 or $(t^2 - r^2)^{1/2}$.

More examples of substitutions can be found in the user's manual [3].

The printing procedure of ORTOCARTAN can handle nested exponentials to an arbitrary height (see Table III in Section 4). More exactly, each line of superscripts can carry one level of its own subscripts, but superscripts to superscripts can go arbitrarily high. The price that we had to pay for this generality is a certain untidiness of the line-breaks. For example, it happens that $f(x)$ is split into two lines as $f(\backslash x)$ or that $\sin^2(x)$ is split as $\sin^2 \backslash(x)$ ($\backslash$ represents here the end of a line). We may correct this in the future, but it is not very easy.

Readers interested in computer science aspects of programming can find more information about the algorithms applied in ORTOCARTAN in Refs. 4–7.

3. TECHNICAL INFORMATION

The newest release of ORTOCARTAN is for Atari computers. The program will run on each model on which the Cambridge LISP dialect of LISP is available. Cambridge LISP is not a property of the authors of ORTOCARTAN and has to be bought separately (see Appendix B). For loading LISP and ORTOCARTAN, 250 kb of core are sufficient, and simple problems could be handled at 300 kb. However, computer-algebra systems are used only on the varieties of Atari with large memories (1 Mb and more). ORTOCARTAN has been implemented on the Mega ST^E model with 4 Mb of core and a 60 Mb hard disk. A hard disk, not necessarily of this large capacity, is essential.

Normally, the output is shown on the screen, but it can also be stored on a disk and printed on paper; the manual [3] explains how to do this. The output stored on a disk can then be processed by a text-editor and built into a text for publication (the tables in this text are examples).

Until recently, the main implementation of ORTOCARTAN was in the University of Texas LISP 4.1 on the CDC Cyber computers (see Refs. 4,5,8).

The code of that version was preserved and is still available on a diskette. However, that version will most probably become defunct because the CDC computers were scrapped in all sites where ORTOCARTAN has been in use on them. The other versions that existed (see Appendix A) are hereby declared defunct.

With 4 Mb of core, the problems with storing large expressions that used to show up occasionally in the past should be gone for ever. A metric whose curvature tensors cannot be stored in 4 Mb will probably have a totally unreadable Ricci tensor, and will anyway require several substitutions to be performed in the intermediate results. One of the more complicated examples processed by ORTOCARTAN was the Kerr solution as represented by Chandrasekhar [9]. It takes the program about $3\frac{1}{2}$ minutes to find that the Ricci tensor is zero in this case, but the list of substitutions takes 4 standard manuscript pages. Even more complicated examples are included in a collection of tests that is distributed on a diskette together with the program.

4. EXAMPLES OF SESSIONS WITH ORTOCARTAN AND CALCULATE

In order to give the reader a better idea of the use of ORTOCARTAN, the input and output for two simple examples are included here.

The program is not fully interactive, i.e. the calculation is not done step-by-step, with user's input being inserted in pieces at intermediate stages. The input has to be defined first, and then the calculation will run all the way until the Weyl tensor. However, the substitutions can be directed to arbitrary intermediate stages of the calculation by simple commands (see user's manual, Ref. 3, for detailed instructions). Thus, effectively, the user has the same control over the run of the calculation as in an interactive system. It is most convenient to prepare the input data on a disk file, the data can then be modified and resubmitted at will.

Table I shows the input data for the Robertson–Walker metric:

$$ds^2 = dt^2 - R^2(t)\,[dr^2/(1-kr^2) + r^2(d\theta^2 + \sin^2\theta\, d\phi^2)]. \qquad (6)$$

For the program, θ was changed to h and ϕ was changed to p, but they could equally well be called *theta* and *phi*, since the symbols do not have to consist of one letter each. The numbers on the right in Table I were added by a text-editor to facilitate the explanation. This whole input was stored on a disk file and read by the LISP system from there. Line (1) is a LISP command that tells the LISP system to treat the upper-case and lower-case

TABLE I
The input data for the Robertson-Walker metric
(the numbers of lines added for explanation only - see text)

```
(setq !*lower nil)                                                    (1)
(ortocartan '(                                                        (2)
   (robertson walker metric)                                          (3)
   (coordinates t r h p)                                              (4)
   (functions R (t))                                                  (5)
   (constants k)                                                      (6)
   (ematrix 1 0 0 0 0 (R / (1 - k * r ** 2) ** (1 2))                 (7)
            0 0 0 0 (R * r) 0 0 0 0 (R * r * (sin h)))                (7)
   (dont print ie agamma gamma riemann)                               (8)
      ))                                                              (9)
(setq !*lower t)                                                     (10)
(rds nil)                                                            (11)
```

letters as different (otherwise, r and R would be considered to represent the same quantity). Line (2) is the call to the function ORTOCARTAN that performs the calculation. Line (3) is the title of the problem. This item is completely arbitrary and serves only to label the input and output with a label chosen by the user. Line (4) defines the names of coordinates. Line (5) defines the names of arbitrary (unknown) functions and their explicit arguments. Line (6) defines the names of constants. Lines marked by (7) specify the components of the orthonormal tetrad $e^i = e^i{}_\alpha dx^\alpha$ which is in this case $e^0 = dt$, $e^1 = R(1-kr^2)^{-1/2}dr$, $e^2 = rR\,d\theta$, $e^3 = rR\sin\theta\,d\phi$. Line (8) tells the program not to print the components of the inverse tetrad, the antisymmetrized Ricci rotation coefficients $\Gamma^i{}_{[jk]}$, the full Ricci rotation coefficients $\Gamma^i{}_{jk}$ and the scalar components of the Riemann tensor. Those parts of the printout are suppressed in order to make Table II shorter. Line (9) contains the two right parentheses that close those of line (2). The items labelled by the numbers 3 to 8 can extend over several lines each, if necessary, and the items (4) to (8) can be written in an arbitrary order. Line (10) undoes the command from line (1), and line (11) tells the LISP system to revert to the keyboard for more input.

Table II shows the complete printout for the input data from Table I. Table III shows the input and output for a simple application of CALCULATE. It is meant mainly to demonstrate the abilities of our printing procedure. With reference to Table I, it should be self-explanatory.[6] The texts in the

[6] In Tables II and III a text-editor was used to remove several empty lines in order to make the tables smaller.

tables are scanned copies of the authentic input and output.

TABLE II
The output for the data from Table I

```
(robertson walker metric)

          0
>   ematrix  . = 1
          0

          1                2 - (1/2)
>   ematrix  . = R (1 - k r )
          1

          2
>   ematrix  . = r R
          2

          3
>   ematrix  . = r R sin (h)
          3

(ematrix completed)
(TIME = 0 msec)

                            2  3         2 - (1/2)
>   DETERMINANT EMATRIX = r  R  (1 - k r )        sin (h)

(DETERMINANT EMATRIX calculated)
(TIME = 0 msec)

(ie calculated)
(TIME = 0 msec)

(agamma calculated)
(TIME = 2000 msec)

(agamma completed)
(TIME = 2000 msec)

(gamma calculated)
(TIME = 2000 msec)

(gamma completed)
(TIME = 2000 msec)

(riemann calculated)
(TIME = 4000 msec)
```

```
(riemann completed)
(TIME = 6000 msec)

                      -1
>   ricci     = - 3 R   R,
        0 0             t t

                     -2       -2     2    -1
>   ricci     = 2 k R   + 2 R   R,    + R   R,
        1 1                        t          t t

                     -2       -2     2    -1
>   ricci     = 2 k R   + 2 R   R,    + R   R,
        2 2                        t          t t

                     -2       -2     2    -1
>   ricci     = 2 k R   + 2 R   R,    + R   R,
        3 3                        t          t t

(ricci calculated)
(TIME = 6000 msec)

(CURVATURE INVARIANT calculated)
(TIME = 6000 msec)

                                    -2       -2     2       -1
>   CURVATURE INVARIANT = - 6 k R   - 6 R   R,    - 6 R   R,
                                              t             t t

(weyl calculated)
(TIME = 6000 msec)

(ALL COMPONENTS OF THE WEYL TENSOR ARE ZERO)

(I REALLY LIKED THIS! CAN I HAVE MORE ?   PLEASE ?!?)

END OF WORK
(RUN TIME = 6000 msec)
```

TABLE III
The input and output for a simple test of CALCULATE

```
(calculate '(
   (print example)
   (constants a b c d)
   (coordinates x)
   (functions f (x))
   (operation (deriv x (a ** (b ** ((der x f) ** (c ** d))))))
      ))

(rds nil)

(print example)

(I UNDERSTAND YOU REQUEST THE FOLLOWING EXPRESSION TO BE SIMPLIFIED)

                        d
                       c
                   f,
                     x
                   b
>       deriv (x,a      )


THE RESULT IS

                      d
                     c        d
                  f,          c                                        d
                    x
                 b        f,       d                                - 1 + c
                            x
>   result  = a         b        c  log (a) log (b) f,      f,
          1                                          x x     x

(I REALLY LIKED THIS ! CAN I HAVE MORE ? PLEASE ?!?)

END OF WORK
(RUN TIME = 2000 msec)
```

APPENDIX A. A SHORT HISTORY OF ORTOCARTAN

ORTOCARTAN was created by an informal team whose members never met all together. This appendix gives credit to everyone who significantly contributed to the development of the program. The contributors are listed in chronological order of entry into the team:

1. Mateusz Wardęcki wrote the algebraic simplification program as part of his M. S. thesis at the Department of Mathematics and Mechanics of the Warsaw University in 1973 [10]. The program was intended for the Polish computer KAR65, but was never implemented because the small core-size of the computer did not allow for practical work with algebraic computing. Later, the program reached the hands of M. Perkowski and became the heart of ORTOCARTAN.

2. Marek Perkowski (then at the Warsaw Technical University, now at Portland State University, USA), in collaboration with this author (A. K.), put together the first complete code of ORTOCARTAN in 1977, and then assisted A. K. in learning LISP and the tasks described in point 3. In writing our program, we were inspired by the program CLAM [11] that we were using then for a brief period. We deliberately reproduced some of the features of CLAM's input syntax and output style.

3. A. K. installed ORTOCARTAN on the CDC Cyber 73 computer in Warsaw in 1977, then debugged and optimized it.

4. Zdzisław Otwinowski (then a student of physics at Warsaw University, now somewhere in Chicago) entered the team in 1979 and contributed several brilliant ideas that resulted in making the algorithms simpler, more general and more efficient. His contributions resulted, among others, in reducing the execution times of the program by the factor of 5 to 10 (depending on the problem).

5. Jacques Richer and Arthur Norman (at Cambridge University) modified ORTOCARTAN in 1981 to run with Cambridge LISP on the IBM 360/370 computers and their compatible German copies; that version is now defunct.

6. A. K. extended ORTOCARTAN for a few "abacus" programs (CALCULATE was one of them). The work was part of a project supported by the Alexander von Humboldt Foundation and hosted by J. Ehlers at the Max Planck Institute in Garching in 1981–82. In 1983, A. K. rewrote ORTOCARTAN into SLISP/360 and installed it on a Siemens 4004 computer at Konstanz University. That task was part of a project supported by the Deutsche Forschungsgemeinschaft and hosted by H. Dehnen; the work was assisted through mail-exchange by J. Fitch. That version is now defunct, too.

7. Goktürk Üçoluk and E. Karabudak rewrote ORTOCARTAN into LISP 1108 and installed it on a UNIVAC 1100 computer at the Istanbul University in 1982. It was reportedly of some use there, but the version is most probably defunct now.

8. A. K. extended ORTOCARTAN for substitutions by pattern-matching and for a few other conveniences in 1984. It was a project supported by the Deutsche Forschungsgemeinschaft and hosted by F. Hehl at the University of Cologne.

9. The new Cambridge LISP version was created from the 1984 U.T. LISP extension and the old (1981) Cambridge LISP version. It is fully equivalent to the latest U.T. LISP version and takes over as the main implementation.

APPENDIX B. HOW TO OBTAIN ORTOCARTAN

ORTOCARTAN can be used only together with Cambridge LISP. Information on this dialect of LISP can be obtained from:

Professor John Fitch, Director
CODEMIST Limited
"Alta", Horsecombe Vale
Combe Down
Bath, Avon, BA2 SQR
England

All information concerning ORTOCARTAN itself can be obtained from the author of this article. The main program comes together with CALCULATE and the user's manual [3] on a single diskette. The manual provides technical information on installing Cambridge LISP and ORTOCARTAN on Atari computers. Should this be insufficient, users are welcome to contact this author.

REFERENCES

1. Frick, I. (1977). "SHEEP—user's guide." University of Stockholm Report 77-15.
2. MacCallum, M. A. H. (1984). In *Classical general relativity. Proc. Conf. on Classical (non-quantum) General Relativity (London 1983)*, W. B. Bonnor, J. N. Islam, M. A. H. MacCallum, eds. (Cambridge University Press, Cambridge).
3. Krasiński, A., Perkowski, M. (1992). *The system* ORTOCARTAN*—user's manual* (4th ed., Warsaw, available on diskette only).
4. Krasiński, A. (1983). *SIGSAM Bulletin* **17**, 12.
5. Krasiński, A. (1986). In *International Conference on Computer Algebra and its Applications in Theoretical Physics*, N. N. Govorun, ed. (Joint Institute for Nuclear Research, Dubna).

6. Krasiński, A. (1991). In *Computer Algebra in Physical Research*, D. V. Shirkov, V. A. Rostovtsev and V. P. Gerdt, eds. (World Scientific, Singapore).
7. Krasiński, A., Perkowski, M., Otwinowski, Z., Kwaśniewski, M. (1984). *The system* ORTOCARTAN *for analytic calculations, detailed description* (2nd ed., Warsaw, available on diskette).
8. Krasiński, A., Perkowski, M. (1981). *Gen. Rel. Grav.* **13**, 67.
9. Chandrasekhar, S. (1979). In *General Relativity: An Einstein Centenary Survey*, S. W. Hawking and W. Israel, eds. (Cambridge University Press, Cambridge).
10. Wardęcki, M. (1973). "Some heuristic methods of symbolic integration" M. S. thesis (in Polish), Department of Mathematics and Mechanics, Warsaw University.
11. d'Inverno, R. A. (1975). *Gen. Rel. Grav.* **6**, 567.