THE SYSTEM ORTOCARTAN USER'S MANUAL

Andrzej Krasiński
N. Copernicus Astronomical Center, Polish Academy of Sciences
Bartycka 18, 00-716 Warszawa, Poland
email: akr@camk.edu.pl

and

Marek Perkowski
Department of Electrical Engineering, Portland State University
P. O. Box 751, Portland, Oregon 97 207, U. S. A.
email: mperkows@ee.pdx.edu

FIFTH EDITION
Warszawa, April 2000

Contents

1 Application of the program.

The program Ortocartan is devised for automatic calculation of the curvature tensors and some related quantities in general relativity from a given orthonormal-tetrad representation of the metric tensor. It was originally written in the University of Texas Lisp 4.1 programming language and implemented on a CDC Cyber 73 computer. That version, and a few later ones, went out of operation together with the computers on which they were implemented. As for today, the only existing versions are 1. In Cambridge Lisp on the Atari Mega STE computers and 2. In Codemist Standard Lisp for the Windows 98 and Linux operating systems. The latter is now the main version, and this description will deal only with this one. (An older edition of this manual is available for the Cambridge Lisp version.) Additional programs, based on Ortocartan, that perform other kinds of calculation, are described in the Appendices B and C.

Please send all correspondence concerning the program to A. Krasiński.

2 The algorithm modelled by the program.

The input data for the program is the tetrad of differential forms:

$$e^i \stackrel{\text{def}}{=} e^i{}_\alpha dx^\alpha \tag{1}$$

$i = 0, 1, 2, 3, \alpha = 0, 1, 2, 3$, summation over all the values of a repeated index is implied. The forms e^i represent the metric tensor according to the formula:

$$g_{\alpha\beta} dx^\alpha dx^\beta = \eta_{ij} e^i e^j \tag{2}$$

where η_{ij} is assumed to be the matrix

$$\eta_{ij} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{3}$$

(i.e. the tetrad e^i is orthonormal).

The program calculates the determinant of the matrix $e^i{}_\alpha$, the inverse matrix $e^\alpha{}_i$, the antisymmetrized parts of the Ricci rotation coefficients $\Gamma^i{}_{[jk]}$ defined by

$$de^i = \Gamma^i{}_{[jk]} e^j \wedge e^k, \tag{4}$$

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

$$\Gamma_{ijk} = \Gamma_{i[jk]} - \Gamma_{j[ik]} - \Gamma_{k[ij]}, \tag{5}$$

where

$$\Gamma_{i[jk]} = \eta_{is} \Gamma^s{}_{[jk]}, \tag{6}$$

and

$$\Gamma^i{}_{jk} = \eta^{is}\Gamma_{sjk} \tag{7}$$

The matrix η^{ij} is the inverse matrix to η_{ij}, numerically identical to η_{ij}.

Further, the program calculates the tetrad components of the Riemann tensor, R_{ijkl} , defined by:

$$d\Gamma^i{}_j + \Gamma^i{}_s \wedge \Gamma^s{}_j = (1/2)R^i{}_{jkl}e^k \wedge e^l, \tag{8}$$

where

$$\Gamma^i{}_j = \Gamma^i{}_{jk}e^k, \tag{9}$$

$$R_{ijkl} = \eta_{is}R^s{}_{jkl}, \tag{10}$$

the tetrad components of the Ricci tensor, R_{ij}, defined by:

$$R_{ij} = R^s{}_{isj}, \tag{11}$$

the scalar curvature R defined by:

$$R = \eta^{ij}R_{ij}, \tag{12}$$

and finally the tetrad components of the Weyl tensor, C_{ijkl}, defined by:

$$C^{ij}{}_{kl} = R^{ij}{}_{kl} + (1/2)\delta^{ijr}_{kls}R^s{}_r - (1/3)\delta^{ij}_{kl}R, \tag{13}$$

where

$$C_{ijkl} = \eta_{ir}\eta_{js}C^{rs}{}_{kl}, \tag{14}$$

and δ^{ijr}_{kls} and δ^{ij}_{kl} are multiple Kronecker deltas defined by the following properties:

1. $\delta^{\cdots}_{\cdots}$ is equal to +1 when none of the values of the upper indices is repeated, while the set of lower indices is an even permutation of the upper ones.

2. $\delta^{\cdots}_{\cdots}$ is equal to -1 when none of the upper indices is repeated, while the set of lower indices is an odd permutation of the upper ones.

2. $\delta^{\cdots}_{\cdots} = 0$ in all other cases, i.e. if either any of the values of upper or lower indices is repeated or if the lower indices are not just a permutation of the upper ones.

The quantities defined above are calculated in every run of the program. However, on special request of the user, the program may also calculate the tetrad components of the Einstein tensor defined by:

$$G_{ij} = R_{ij} - (1/2)\eta_{ij}R, \tag{15}$$

as well as the coordinate components of all the quantities, including the metric tensor and Christoffel symbols. With the exception of the Christoffel symbols for which the valence is fixed, the indices of the tensor components may be in any desired positions, e.g. for the Riemann tensor one may obtain $R_{\alpha\beta\gamma\delta}$ and $R^\alpha{}_{\beta\gamma\delta}$ and $R^{\alpha\beta}{}_{\gamma\delta}$, and so on. The tensor components are calculated as secondary objects by contractions of the sets of tetrad

components of the appropriate quantity with the tetrad vectors given on input, or with the inverse tetrad vectors found in the first step. For example:

$$R_{\alpha\beta\gamma\delta} = e^{i}{}_{\alpha} e^{j}{}_{\beta} e^{k}{}_{\gamma} e^{l}{}_{\delta} R_{ijkl} \tag{16}$$

$$R^{\alpha}{}_{\beta}{}^{\gamma}{}_{\delta} = e^{\alpha i} e^{j}{}_{\beta} e^{\gamma k} e^{l}{}_{\delta} R_{ijkl} \tag{17}$$

Just how these, and some other additional requests of the user may be communicated to the program, will be explained further.

To use the program one does not need any special knowledge. We only expect our users to be familiar with simple text-editing and copying files. All the remaining tiny amounts of information will be supplied by the present text. In the first place, the user must know two elementary notions of Lisp: the atom and the list.

3 Atoms.

The definition of an atom given below is, with respect to the general Lisp definiton, a restricted one so that it fits the needs of Ortocartan. An atom is either an integer number or a continuous string of up to 72 characters[1] which does not begin with a number, and does not contain any of the following special characters:

NAME	PRINT NAME
left parenthesis	(
right parenthesis	)
period	.
comma	,
blank	
dollar sign	$
colon	:
backslash	\
slash	/
exponentiation sign	^
multiplication sign	*
plus or minus	+ or -

The user is warned that different keyboards are not necessarily compatible, and various special signs on them often correspond to different symbols. For safety, the user is therefore advised to avoid special signs like #, [,] or & unless one is aware what one really does with them. Any other characters available on the input equipment are permissible. Each of the prohibited characters has a special meaning for the Lisp system and, when used inside a string of characters, would cause splitting the string into more atoms or other structures.

Examples of atoms:

[1]The Codemist Standard Lisp has 72 characters per line on the screen. Longer atoms may be tolerable within the program code, but they would look untidy when showed on the screen.

`A 12 DELTA X1 ATOM`

Examples of strings which are not atoms:

`(B) A,BC AT.L 1.5 DOLLAR\$` - each of these contains forbidden characters,

2DELTA - this one starts with a number

Floating-point numbers, which are atoms for the Lisp-system, are not used in precise calculations as their use introduces automatically decimal approximations of fractional numbers, very much unwanted here. Therefore floating-point numbers are illegal in Ortocartan.

4 Lists.

A list is a string of characters starting with the left parenthesis, ending with the right parenthesis, and containing atoms or other lists separated one from the other by single or multiple blanks or by single commas. Many consecutive blanks have the same meaning as a single blank. Two or more consecutive commas or commas separated only by blanks form, in some Lisp systems, an illegal character set, and they result in an error. Blanks are not necessary (but allowed) in front of and behind a parenthesis. Examples of lists:

```
(      )+ - this is an empty list, having no elements
(ATOM)
(SOME MORE ATOMS)
((QUITE) (A (COMPLICATED)(((LIST)))))
```

Examples of strings which are not lists:

(SOMETHING (((IS) MISSING)) - one right parenthesis missing.

This is a very common type of error and one should be careful to avoid it. It leads to unpredictable error-messages that can be very misleading.

(AGAIN ((SOMETHING (MISSING))))) - one left parenthesis missing or one extra right parenthesis.

(WHAT) (IS THIS) - this is a series of two lists

Atoms and lists together are called "symbolic expressions" or "S-expressions".

5 The representation of mathematical formulae in Ortocartan.

The notation required for Ortocartan is similar to the conventional mathematical notation. The restrictions or changes result from the fact that the mathematical notation is in some points nonunique, and is intelligible uniquely only when accompanied by some explanatory text, which, of course, cannot be supplied to the computer in the form of English phrases.

Sums are represented exactly as in mathematics. One can omit the "+" sign before the first term of a sum or insert it - both cases are legal. One must only be careful to separate the "+" and "-" signs from neighbouring terms by blanks or parentheses, so that they form separate atoms. Also when a sum is too long to fit into one line of input, and must be continued in the next line, then the "+" or "-" sign may be placed either at the end of the preceding line, or at the beginning of the following line, but not in both these places simultaneously as this would cause an error (the second sign would be understood as a symbol of some quantity).

Products are represented as in mathematics, with the restriction that the multiplication sign, *, must not be omitted, and must be separated by blanks or parentheses from neighbouring factors.

Exponentiations are represented differently, because writing them in coupled parallel lines would cause hard problems for the Lisp input procedure. So exponentiation is written in one-line format in the following form:

```
<the base> ^ <the exponent>
```

just like in FORTRAN. For instance, A^B will be represented by (A ^ B). In some versions of Lisp, the symbol ^ automatically splits any sequence of characters into separate atoms. In those versions the symbol A^B would be understood as the sequence of 3 atoms, the A, the ^ and the B. However, in other versions of Lisp, A^B would be understood as a single atom consisting of the 3 characters. Therefore, it is always safer to separate all signs of mathematical operations from their arguments by blanks, like in all examples in this manual.

Division is denoted by the slash /, (A / B) representing A / B.

Ortocartan uses the same hierarchy of algebraic operations as is used in ordinary algebra, i. e. the algebraic operations are performed in the following sequence: first exponentiations, then multiplications, then divisions, then additions with subtractions. This order is changed by the use of parentheses, just like in algebra.

Negative integers may be represented either as single atoms whose first character is the sign "-", e.g. -2 or -10, or as a list of two atoms where the first atom is the sign "-", and the second atom is the absolute value of the number, e.g. (- 2) or (- 10). Both representations are correct. For non-numerical quantities however, the second representation is the only correct one: (- A) means "the negative of A", while -A, depending on the Lisp system, would be understood either as a sequence of two atoms or as a single atom which would be treated by the program Ortocartan as a single symbol of some quantity.

Non-integer rational numbers are represented as two-element lists, the first element being the numerator, and the second element the denominator of a fraction, e.g. 1/2 will

be represented as (1 2). Negative fractions may be also represented in two ways, e.g. -2/3 may be written as (-2 3) or as (-(2 3)). Both representations are correct, but the first one is recommended.

Functional expressions must be represented as lists, the first element of the list being the name of the function, and the following elements being consecutive arguments of that function. Note: each argument (or the argument if it is single) of a function must be a single S-expression, i.e. either a single atom or a single list. For example, the correct representation of sin(xy) is (sin(X * Y)), while (sin X * Y) would be understood as a functional expression in which the function sin is given three arguments: X, *, and Y. Of course, such a three-argument sin would be processed by the program quite incorrectly.

The program "knows" and can process automatically the following functions: exp, log (which stands, of course, for natural logarithms to base e), cos, sin, tan, ctan, cosh, sinh, tanh, ctanh, arctan, arcsin, arsh (the function inverse to sinh), arch (inverse to cosh), and arth (inverse to tanh). All these functions are correctly differentiated, and the most important simplifications, like (exp (log X)) = X, (log (X * Y)) = (log X) + (log Y), (tan X) * (ctan X) = 1 or (sin X) ^ 2 + (cos X) ^ 2 = 1 are made on them.

If it is necessary to introduce a symbol for a derivative of some function of unspecified form, like dF/dx, then the derivative should be written as `(der <the variable> <the function>)`. In this example, one should write (der X F). Multiple derivatives require only inserting the full series of variables between der and the name of the function, e.g. $F_{,xxy}$ should be written as (der X X Y F) or (der X Y X F) or (der Y X X F) (The program is not sensitive to such subtleties as commutativity of derivatives or even differentiability of the functions. It just assumes that all the functions to be differentiated are differentiable, and that in multiple derivatives all the differentiations commute). Partial derivatives are denoted by the same symbol as total derivatives.

Sometimes it is hard to calculate by hand an explicit derivative of a large expression, which must be inserted into the data. In this case the task of calculating the derivative may be left to the program. The format for writing such a derivative is the same as above, only instead of the atom der one should write the atom deriv. For instance, suppose one is too lazy to calculate explicitly the expression:

$$(d/dx)(\sin^2 x + ax \sin^4 x + ax \log x \sin x)$$

but such a derivative must be inserted into the data. Then one should write:

```
(deriv X ((sin X) + A ^ 2 * X * (sin X) + A ^ 4 * X * (log X) * (sin X)))
```

and the program will do the work. Actually, der may be always substituted by deriv, and it is even advisable to do so in case of multiple partial derivatives. In this case, the series of variables should be properly ordered for uniqueness. The deriv-functional expression will be ordered, while der will be left intact literally as given by the user. For instance, if the user writes (der X Y X F), then such an expression will not be changed, while if one writes (deriv X Y X F), then such an expression will be changed either to (der X X Y F) or to (der Y X X F) depending if X or Y has the higher priority.

An indefinite integral should be represented as `(int <the variable> <the integrand>)`. For example $\int G dx$ should be represented as (int x G), and $\int e^{\sin^2 x} dx$ should be represented as (int x (exp((sin x)^ 2))).

The program is not able to process definite integrals. If a definite integral must be introduced, then it can be declared as a function of unspecified shape whose derivatives should be substituted by the required expressions supplied by the user on input (how such substitutions may be requested - see further).

Let us now give a few examples of more complicated expressions represented on input for the program Ortocartan:

$(1/3)Lr^2(r^2 + a^2) + r^2 - 2mr + a^2$ should be represented as:

```
((1 3) * L * r ^ 2 * (r ^ 2 + a ^ 2) + r ^ 2 - 2 * m * r + a ^ 2)
```

$-1/(1 - 2m/r)$ should be represented as (-1 / (1 - 2 * m / r)), or as (-(1 / (1 - 2 * m / r))), or as `(-(1 - 2 * m / r) ^ -1)`.

$a(df/dx) + \int (x/V)dx$ should be represented as:

(a * (der x f) + (int x (x / V))).

The program prints the results in the mathematical format, i.e. with superscripts, subscripts and exponents all in their proper places. Derivatives are printed according to the common convention in general relativity: the name of the function is followed by the comma and by subscripts being the names of the variables of differentiation. For example, dF/dx, written on input as (der x F) will be printed as $F_{,x}$, while $\partial^3 G/\partial x \partial y^2$ will be printed as $G_{,xyy}$.

Unfortunately, most Lisp systems do not support sophisticated text-editing and cannot print more elaborate signs like the integral. Therefore indefinite integrals are printed similarly as they stand in the input, e. g. $\int F dx$ will be printed as int (x F). (See, however, section 23: in Ortocartan and in the associated programs one can write the output on a disk in the form of Latex code, and then the printout may be passed through Latex, with more satisfactory results in some cases.)

6 Typing the input.

The whole input described in the sections that will follow can be typed in directly from the keyboard. However, it will often contain misprints or will require adding new substitutions and resubmitting for calculation. Therefore the user will usually want to prepare an input file and let Ortocartan read the data from there. How to write the data is described in the next section. You can save the data file using any text-editor. When you want Lisp to read the data from the file, you have to write, while working within Lisp:

```
(rdf '"<the name of the file>")
```

where `<the name of the file>` should include the whole path of access, unless the data file is in the same directory as the Lisp core-image. Remember to write the apostrophe – it means that Lisp should not look for a value of `"<the name of the file>"`, but just take the name literally. The quotation marks are necessary in order that Lisp tolerates untypical characters like the backslashes \ or slashes / or dots. Without the quotation marks, each such untypical sign would have to be preceded by the exclamation mark. For example, this is how I myself ask Lisp to read data for the Robertson-Walker metric (to be included in this manual further on) from a disk file:

```
(rdf '"\akr\ortocar\robwal.dat")
```

After reading such a command, Lisp will read the file `\akr\ortocar\robwal.dat` and carry out whatever task it was given in those data. Then it will automatically go back to the keyboard for more input.

When typing the input directly from the keyboard, be careful not to continue any atom across the right margin - the Lisp system might split one of the next atoms into two. If an atom seems too long to fit into the current line, then simply press "Return" (or "Enter"). The program will not start running until you close all parentheses and press "Return" after the last one.

7 Starting Lisp and Ortocartan.

Codemist Standard Lisp must be bought from its owners, the Codemist Limited (see Appendix A). Installing it on your computer will most probably require the assistance of your computer staff. We assume that you have the Lisp already at your disposal and describe how to load the Ortocartan programs into it.

Since the system Ortocartan will be written into the computer's core on top of Lisp as a core-image, make sure first that you have a second copy of the pure Lisp core-image, to which you can revert if you wish to use Lisp without Ortocartan.

The diskette with the Ortocartan programs contains, among other things, the following files:

1. Ortcsl.lis, containing the program for calculating the curvature tensors as described in Section 2. This is the main program, and all the other programs make use of parts of this one. The main program can be used alone, the other programs must be loaded into core only together with the main one. How to use the other programs is described in the Appendices B and C.

2. Calcsl.lis - this is the "algebraic abacus" program.

3. Elliscsl.lis. This is the program that calculates the Ellis evolution equations.

4. Ncurvcsl.lis. This is the program for calculating the curvature tensor for given connection coefficients in an arbitrary number of dimensions.

5. Lanlagcs.lis - the program for calculating the lagrangian by the Landau-Lifshitz method for a given metric.

6. Eulagcsl.lis - the program for calculating the Euler-Lagrange equations from a given lagrangian.

7. Squintcs.lis - the program for checking the first integrals quadratic in first derivatives.

The files *.tes contain sets of test examples for the corresponding programs.

The sequence of actions is now the following. Copy all the *.lis files onto your disk. Let the directory where the files are stored have the name

\akr\ortocar

(this is my directory - you can choose any name you like). Call Lisp and then, from within Lisp, write:

```
(rdf '"\akr\ortocar\ortcsl.lis")
(rdf '"\akr\ortocar\calcsl.lis")
(rdf '"\akr\ortocar\elliscsl.lis")
(rdf '"\akr\ortocar\ncurvcsl.lis")
(rdf '"\akr\ortocar\lanlagcs.lis")
(rdf '"\akr\ortocar\eulagcsl.lis")
(rdf '"\akr\ortocar\squintcs.lis")
```

Lisp will respond to each (rdf ...) by confirming that it has defined several functions whose names and meanings need not bother you. Then write:

(reclaim)

This command will cause that the Lisp system will clear the core of all useless data. Next write:

(preserve)

After this command Lisp will write the core-image to the disk file and quit. Next time when you call Lisp from the same image, all the Ortocartan definitions will already be parts of the Lisp system.

At this point, you do not know yet how to use the program Ortocartan. This will be described in the sections that follow. The thing to remember now is this:

Codemist Standard Lisp (the same is true for Cambridge Lisp and for at least some other Lisp systems) can either consider upper case letters identical to their lower case counterparts (this is the default mode), or can treat them as different symbols. If you wish the upper- and lower case letters to be considered different, then write:

(setq !*lower nil)

From now on, however, be careful to write all commands in lower case letters or else Lisp will not recognize them. In particular, you should write `(rdf <filename>)`; if you write (RDF ....), then Lisp will protest that the function RDF is undefined. When you wish to give up this additional convenience, write:

(setq !*lower t)

When you wish to finish the session with Lisp, click on the cross in the upper-right corner of the screen.

Should you wish to preserve the Lisp output on a disk, click the "File" in the upper-left corner of the screen, and then click "To file" in the dropdown menu. Then you will be guided by the icons so that you can choose the directory where the output should be written. When you wish to stop writing the output to the file, click "File" again and then click "Terminate log". Note: if you choose to write anything to the same file again after you have "terminated", the old contents of the file will be overwritten with the new data.

8 The dictionary for communicating with the program.

For the purposes of communicating the user's requests to the program, all of the quantities that the program can calculate were assigned their unique names. The dictionary of those names is given in the table below.

STANDARD NAME OF THE QUANTITY	PROGRAM NAMES	
	TETRAD COMPONENTS	TENSOR COMPONENTS
metric tensor and its inverse	–	g
the array of coefficients of the tetrad of forms $e^i{}_\alpha$	ematrix	
determinant of the matrix $e^i{}_\alpha$	determinant	
the array of coefficients of the inverse tetrad $e^\alpha{}_i$	ie	
antisymmetrized Ricci rotation coefficients	agamma	–
full Ricci rotation coefficients	gamma	–
Christoffel symbols	–	christoffel
Riemann tensor	riemann	rie
Ricci tensor	ricci	ric
scalar curvature	curvature	
Einstein tensor	einstein	ein
Weyl tensor	weyl	c

The program names are easy to remember because, with two exceptions, they are the same as the names that appear on output. The exceptions are: the "determinant" (appears on output as "determinant ematrix"), and the "curvature" (appears on output as "curvature invariant").

9 The printout.

All the results of the program are printed in the mathematical many-line format. They are printed in the form of equations:

```
<name of the quantity> <indices attached to the name  (sometimes  none)>  =
<value of the appropriate component of the quantity>
```

The tetrad components of the Riemann, Ricci, Weyl and Einstein tensors are printed with all indices down. The tensor components (if requested by the user to be calculated) are printed with the appropriate positions of the indices, corresponding to the valence requested. The tetrad and tensor components of the same quantity are printed under different names (see the table in section 8) so that one cannot get confused about distinguishing them. However, the tetrad $e^i{}_\alpha$ and the inverse tetrad $e^\alpha{}_i$ have each one tetrad- and one tensor-index. To denote which is which, a dot is placed above or below the tensorial index in print. For instance, the component $e^0{}_0$ of the matrix $e^i{}_\alpha$ will be printed as:

```
       0
EMATRIX  . = <appropriate expression>
         0
```

while the component $e^0{}_0$ of the inverse matrix $e^\alpha{}_i$ will be printed as:

```
  0
IE.   = <appropriate expression> .
    0
```

The program follows a general convention used in relativity theory: for all the quantities only their independent nonzero components are printed. For instance, advantage is taken of the symmetry of the Ricci tensor, and of the components R_{ij} only those with $i \leq j$ will appear (if nonzero) in print. The same is true for the Riemann tensor: of the components R_{ijkl} only those will be printed for which simultaneously $\{i < j\}, \{i \leq k < l\}$ and $\{j \leq l$ while $i = k\}$, and whose value is nonzero. If for some quantity all the components happen to be zero, then, in order to avoid the user's confusion, the message is printed:

```
ALL COMPONENTS OF THE <name of the quantity> ARE ZERO.
```

Now if all the Ricci rotation coefficients turn out to be zero, then everyone can guess that the Riemann tensor and all its concommitants will also be zero. In that case the program stops after printing the message:

(ALL COMPONENTS OF THE AGAMMA ARE ZERO).

If the Riemann tensor turns out to be zero, then again it makes no sense to go on calculating the other quantities which are all zero, so the work is stopped after printing the appropriate message. If the Ricci tensor turns out to be zero, then the Weyl tensor will be just equal to the Riemann tensor, so also in this case the work is stopped after printing the message about the Ricci tensor being zero.

The message END OF WORK marks the proper end of the printout unless an error occured during the calculation (see below).

The printout are the results of the calculation, interspersed with the time-messages of the form:

```
(ematrix completed)
(TIME =  <n1>  msec)

(DETERMINANT EMATRIX calculated)
(TIME =  <n2>  msec)

(ie calculated)
(TIME = <n3>  msec)

(agamma calculated)
(TIME = <n4>  msec)

(gamma completed)
(TIME = <n5>  msec)
```

and so on, where `< n1 > to < n5 >` are times in milliseconds elapsed from the start of the work on the current problem until completing the appropriate step of the calculation. A quantity is calculated when all its independent components were found, and it is completed when also the dependent components were found with use of the symmetry properties. The time-messages are not always useful and they can be suppressed (see sec. 20).

If an error occurred during the calculation, then the printout is just suddenly terminated and followed by an error-message. If no error occurred, then the end of the printout is marked by the message END OF WORK, followed by the information `(RUN TIME = <n> msec)`.

Note: All the input data are processed by the procedure of algebraic simplification, so that they acquire the standard form required by the program. They are reprinted in the mathematical format after they are standardized. Therefore, they may differ from the original data by ordering the sums and products or by the replacement of $\cos^2 x$ by $1-\sin^2 x$, and so on. You should study this part of the printout carefully because it shows what the program is really processing.

10 The frame for the data.

The first and the last item of the input data are always the same, and for clarity are best placed in a separate line each. The first item has the form:

(ortocartan '(

(do not overlook the apostrophe!).

The second item of the data is the title of the problem. It is a completely arbitrary single S-expression (i.e. either a single atom or a single list). The precise shape of the title has no meaning for the program, it is simply printed as the heading of the printout in order to mark it by an easily recognizable identifier chosen by the user. For example, here are some possible titles:

CURVATURES
(SCHWARZSCHILD METRIC)
(GENERAL SPHERICAL METRIC)

Note however that the title:

AXIAL METRIC

would result in an error: it consists of two S-expressions. In this case the program would understand that the atom AXIAL is the title while the atom METRIC is a part of the relevant data (see further sections). Soon it would recognize that the atom METRIC has none of the required forms for the data, and would print the message:

```
******  ///  METRIC  ///  *****  IS AN ILLEGAL ARGUMENT FOR OUR SYSTEM. SORRY,
CANT GO ON.
```

A convenient title is a reference to the paper from which the metric was taken.

The last item of the data is the set of two right parentheses:

))

They close the two parentheses from the first line.

All the other items of the data, described in the sections that follow, should be placed between the title and the last line with parentheses, in an arbitrary order.

11 Declaration of coordinates.

The user tells the program the names he has chosen for the coordinates by inserting into the data the item of the form:

```
(coordinates <atomic names for the coordinates> ).
```

The expected order is x^0, x^1, x^2, x^3 where x^0 is the time coordinate. Examples:

```
(coordinates  X0  X1  X2  X3)
(coordinates  T   X   Y   Z )
(coordinates  t   r   theta   phi)
(coordinates  T   PH1  X1   X2)
```

There are no default names for non-declared coordinates, so this piece of the data cannot be omitted. If it is omitted, then Ortocartan will print an error-message and refuse to work.

12 Declaration of arbitrary functions.

It has the following form:

```
(functions <name of function 1> <list of arguments of  function 1>
           <name of function 2> <list of arguments of function 2> .... )
```

For instance, if the user wants to use the functions $f(x, y, z)$ and $g(x, y)$, then the appropriate declaration is:

```
(functions f (x y z) g (x y) )
```

If composite functions are to be used, then for each function only those arguments must be given on which it depends directly. The arguments which are themselves functions must be declared on their own. For example, if the user wants to use the functions g(x,y) and f(x,y,h(t,u)), where u is a function of z, then the appropriate declaration is:

```
(functions  g (x y)  f (x y h)  h (t u)  u (z))
```

The order in which the declared functions appear is irrelevant from the point of view of syntax. However, the functions will be ordered by the program in sums and products according to the same order.

If the program Ortocartan is applied to some explicitly given tetrad which does not contain any arbitrary functions, then this piece of data should be simply omitted.

Note: The built-in functions listed in section 5 must not be declared. Of course, their names should not be used for user-defined arbitrary functions. The reserved names, not to be used as the names of functions, include also the atoms nil, times, plus, expt, minus, +, - and *.

In the formulae described in sections 14 and following, only the names of the functions must be written, their arguments do not have to be written out (see examples I - V in Appendix E). However, if the user wishes so, the arguments can be written out. Then

Ortocartan will automatically pick up this style and will write out the arguments of the functions throughout the whole calculation, see Example VI in Appendix E. The derivatives of functional expressions are then printed differently, on the assumption that the arguments are not necessarily atoms, but can be functional expressions themselves. For example, let f be a function of $(x^2 + y^2)^{1/2}$ and z. One can then write in the input data:

```
(f ((x ^ 2 + y ^ 2) ^ (1 2)) z)
```

and the derivatives of F will be printed as:

```
                2    2 -1/2         2    2 1/2
f,  = (1/2) x (x  + y )     f,  ((x  + y )   , z)
  x                           1

                2    2 -1/2         2    2 1/2
f,  = (1/2) y (x  + y )     f,  ((x  + y )   , z)
  y                           1


             2    2 1/2
f,  = f,  ((x  + y )   , z),
  z     2
```

the subscripts on the right-hand sides referring to the consecutive arguments of f. Such an output is less easily readable, however, and we do not recommend this style. The function f from the last example should be declared as being dependent on x, y and z. Note the correct format for writing the functional expression in the input data, it must be a list of the form `(<function name> <argument 1> <argument 2> ... <argument n>)`. The whole functional expression must be enclosed in parentheses, the series of arguments must not be enclosed in an extra pair of parentheses, and the arguments must be separated from each other only by spaces, do not use any commas.

13 Declaration of the constants.

This piece of the data tells the program which symbols should produce zero when differentiated. The declaration has the form:

```
(constants <names of the constants> )
```

For example:

```
(constants M)
(constants KAPPA RHO H)
```

The order of the constants has the same meaning as the order of the functions. If no arbitrary constants appear in the metric, then this piece of data should be omitted.

14 Declaration of the tetrad.

Let us recall that the program Ortocartan can process only orthonormal tetrads in the signature $(+---)$. The tetrad is declared in the form:

```
(ematrix  <components of the matrix e^i_α, in the order

 i=0,A=0 / i=0,A=1 / i=0,A=2 / i=0,A=3 / i=1,A=0 / ... /
 i=3,A=3, and in the notation described in section 5>)
```

For instance, for the Schwarzschild metric:

$$ds^2 = (1 - 2GM/c^2 r)dt^2 - 1/(1 - 2GM/c^2 r)dr^2 - r^2(d\vartheta^2 + \sin^2\vartheta d\varphi^2)$$

the declaration should look as follows:

```
(ematrix ((1 - 2 * G * M / c ^ 2 * r) ^ (1 2))  0  0  0
    0 (1 / (1 - 2 * G * M / c ^ 2 * r) ^ (1 2)) 0 0
    0  0  r  0
    0  0  0 (r * (sin theta))     )
```

Note that each component of $e^i{}_\alpha$ in the above list must be a single S-expression, i.e. must be embraced by parentheses if it is not an atom.

This piece of the data is the heart of the problem, so, needless to say, its omission would push the program into a fatal error. The omission of some relevant parentheses usually causes that the list becomes too long, i.e. has more S-expressions than the expected 17. This kind of error is directly communicated by the system Ortocartan.

The pieces of data described up to this place form a minimal set of data for Ortocartan. This is all if the user has no special requests concerning the results. All the pieces described further serve to adjust the results of the program to the momentary needs of the user.

15 Symbols for sums and other expressions.

In by-hand calculations it is sometimes convenient to develop a product involving a sum by applying the rule of distributivity of multiplication, and sometimes it is not. The choice requires a small piece of intelligent thinking: one must be conscious of the goal one wants to achieve. But intelligence, even in such small amounts, is something (as yet) inaccessible to computers: the program must follow an algorithm which specifies unique decisions or unique criteria of choice. Consequently, in our program the intelligent pieces of work are left to the user himself. The program applies the rule of distributivity always, unless the user requested this not to be done for a specific sum. The format for such a request is described below.

If a sum which is present in the ematrix is not to be expanded when in a product, then the user should introduce a separate symbol for the sum, use the symbol in the ematrix, and insert an additional item into the data, of the form:

```
(symbols <the symbol> = <the sum>)
```

Actually, one can declare in this way an arbitrary number of special symbols for sums:

```
(symbols <symbol 1> = <sum 1>
    <symbol 2> = <sum 2>
    ..................
    <symbol n> = <sum n> )
```

Then, in all the algebraic operations the left-hand sides of the equations will be used, while differentiation will operate on the right-hand sides. Example: take the Nariai metric:

$$ds^2 = \{a(t)\cos[log(r/l)] + b(t)\sin[log(r/l))]\}^2 dt^2 - l^2(dr^2/r^2 + d\vartheta^2 + \sin^2\vartheta d\varphi^2))$$

where l is a constant and a and b are arbitrary functions of time. Suppose, you want the coefficient of dt^2 not to be developed in products, and you call it ψ. Then the declaration should be:

```
(symbols psi = (a * (cos (log (r / l))) + b * (sin (log (r / l)))))
```

Of course, this should be accompanied by:

```
(coordinates t r theta phi)
(constants l)
(functions a(t) b(t))
(ematrix  psi  0  0  0
          0  (l / r)  0  0
          0  0  l  0
          0  0  0  (l * (sin theta)))
```

After such a call to Ortocartan, psi will be used in all the algebraic operations, but for $\partial\psi/\partial t$ and $\partial\psi/\partial r$ the program will automatically substitute, respectively, the quantities $(\partial a/\partial t)\cos[\log(r/l)] + (\partial b/\partial t)\sin[\log(r/l)]$, and $-(a/r)\sin[\log(r/l)] + (b/r)\cos[\log(r/l)]$. This example is shown in full in Appendix E.

The usefulness of symbols is most evident in the case of sums, but the expressions represented by the symbols need not be sums.

The symbols declared on the left-hand sides of the equations should not be declared separately as functions (if they are, this will have practically no result).

The equations listed in the "symbols"-argument are printed in the mathematical format at the very beginning of the printout, preceded by the heading "symbols". Note: Before being printed and applied they are standardized by the simplifying procedure. See therefore if their standardized forms are convenient for you. If not, make the appropriate adjustments in your data.

16 Calculating coordinate components of various quantities.

It is sometimes useful to calculate also (or just) the tensor components of some quantities; e.g. the metric tensor, to check the correctness of the tetrad components, or the Christoffel symbols, to solve the equations of a geodesic. The program Ortocartan can do that: it can calculate the tensorial components of the Riemann, Ricci, Einstein and Weyl tensors, with any required positions of their indices, the metric tensor, the inverse metric, and the Christoffel symbols. In order to have some tensorial components calculated and printed one should insert into the data the item of the form:

```
(tensors
       <name of quantity 1> <valence I required for quantity 1>
                  <valence II required for quantity 1>
                  ..................................
                  <valence  n required for quantity 1>
       <name of quantity 2> <valence I required for quantity 2>
                  ........ )
```

where the `<name of the quantity>` should be the identifier of the quantity to be calculated and printed in its tensorial form. Essentially (with two exceptions) it is the name of

the tetrad-quantity which is the source to calculate the relevant tensor. The dictionary of these names is:

THE TENSOR WANTED	NAME TO BE INSERTED IN (tensors ...)
metric tensor and its inverse	metric
Christoffel symbols	christoffel
Riemann tensor	riemann
Ricci tensor	ricci
Weyl tensor	weyl
Einstein tensor	einstein

`<valence N required for quantity K>` is a list of the signs "+" and "−", the "+" in the i-th position of the list meaning that the i-th index should be an upper one, and the "−" in the j-th position meaning that the j-th index should be a lower one. For instance:

```
(tensors riemann (+ - + -) (+ - - -))
```

means that the user wants the components $R^{\alpha}{}_{\beta}{}^{\gamma}{}_{\delta}$ and $R^{\alpha}{}_{\beta\gamma\delta}$ of the Riemann tensor to be calculated and printed.

If not all indices are on the same level (i.e. are not all contravariant or not all covariant), then the practical use of the symmetry properties may be difficult as lowering or raising an index with a nondiagonal metric often requires much algebra. In this case the symmetries which are not trivial (i.e. do not amount just to equality of some components, or to the equality of a component with the negative of some other component, or to vanishing of a component) are not taken into account, and the program prints the dependent components, too. For instance, when printing $R^{\alpha}{}_{\beta}{}^{\gamma}{}_{\delta}$, only the trivial symmetry $R^{\alpha}{}_{\beta}{}^{\gamma}{}_{\delta} = R^{\gamma}{}_{\delta}{}^{\alpha}{}_{\beta}$ will be used, while the symmetries corresponding to $R_{\alpha\beta\gamma\delta} = -R_{\beta\alpha\gamma\delta}$ and $R_{\alpha\beta\gamma\delta} = -R_{\alpha\beta\delta\gamma}$ will be ignored (i.e. 136 components, if all nonzero, will appear in print). When printing $R^{\alpha}{}_{\beta\gamma\delta}$, only $R^{\alpha}{}_{\beta\gamma\delta} = -R^{\alpha}{}_{\beta\delta\gamma}$ will be used, while $R_{\alpha\beta\gamma\delta} = R_{\gamma\delta\alpha\beta}$ and $R_{\alpha\beta\gamma\delta} = -R_{\beta\alpha\gamma\delta}$ will be ignored (i.e. 96 components, if all nonzero, will appear in print).

One can use the same name repeated a few times, each time followed by a different valence, e.g.:

```
(tensors weyl(- - - -) weyl(+ - - -) weyl(+ + - -) ),
```

but this is equivalent to the simpler expression where the name is used only once:

```
(tensors weyl(- - - -)  (+ - - -)  (+ + - -) ).
```

If a name of some quantity is not followed by a valence-specification, then it is understood that all the indices of the tensor should be down (covariant). There are two exceptions to this:

1. The "christoffel" has an obvious valence, which thus need not be specified.

2. The "einstein" not followed by a valence-specification is understood as a request to calculate the tetrad components of the Einstein tensor (which are not calculated in the routine run of the program).

Each specification of valence (or name of a quantity if no valence is present) may be followed by a series of sets of indices, each set defining a single component of the tensor. Such a form will be understood as a request to calculate only the components thus defined. For instance:

```
(tensors riemann (+ - - -)
    ricci (- -) (+ -) (0 1) (0 2) (1 1)
    weyl (- - - -) (0 1 0 1) )
```

means that the user wants the program to print all the components $R^{\alpha}{}_{\beta\gamma\delta}$ of the Riemann tensor, and all the components $R_{\alpha\beta}$ of the Ricci tensor while of the components $R^{\alpha}{}_{\beta}$ of the Ricci tensor and $C_{\alpha\beta\gamma\delta}$ of the Weyl tensor only $R^0{}_1$, $R^0{}_2$, $R^1{}_1$ and C_{0101} should be printed. The single components specifically requested in this way are printed even if they are equal to zero.

Note that if the symmetries of some quantity (like $R_{\alpha\beta}$) are trivial in practical use, then each set of indices referring to that quantity should be ordered in the proper way, e.g.:

```
(tensors ricci (- -) (1 0) )
```

will result in printing no components of $R_{\alpha\beta}$, as the combination (1 0) of the indices of $R_{\alpha\beta}$ is not considered at all by the program. The correct request is:

```
(tensors ricci (- -) (0 1) ).
```

17 Expanding natural powers of sums.

For purposes of uniqueness, natural powers of sums are automatically expanded with use of rules of the type:

$$(a+b)^2 = a^2 + 2ab + b^2.$$

This, as well as automatic application of the rule of distributivity of multiplication, is necessary to carry out all the algebraic simplifications that are possible. Without this, quite trivial flaws inescapably occur, e.g. the program would not recognize that

$$(1+x)^2 - 1 - 2x - x^2 = 0.$$

However, if the exponent is larger than 3, then the expansion is suspended, and the expression is left in a non-expanded form to avoid too large expressions which would probably

appear in such a case. This might, of course, be inconvenient in some cases, so the user may change this rule, if necessary. In order to do it, one should insert into the data the item of the form:

```
(expand powers from <n1> to <n2>)
```

where <n1> and <n2> are the minimal and the maximal value, respectively, of the exponents with which the expansion should be automatically done everywhere. If $n1 = n2$, then only the single exponent $n = n1 = n2$ will be processed in this way. Any value $n1 < O$ is equivalent to $n1 = O$: all powers up to n2-th will be expanded. If $n2 < O$ or $n2 < n1$, then no powers will be expanded.

If one wants some definite expressions which appear in some definite places to be expanded even if their exponents exceed the upper limit (whether changed before by (expand powers ...) or not), then one should use the facility of substitutions described in the next section.

18 Substitutions.

As already mentioned before, the program cannot be as intelligent as a human being. Consequently, a human being can use much of his (her?) ingenuity to simplify the process of calculation, or the results, while a program will always follow rigid rules, and it will miss some simplifications if they are not of an algebraic nature, or if they involve a calculation with rational functions. For instance, Ortocartan cannot recognize that $\sin x / \cos x = \tan x$ what is sometimes necessary. Therefore, the user may introduce some simplifications by his (her) own. The request to make such user-defined substitutions is expressed by inserting into the data the item of the form:

```
(substitutions  <a series of addresses> <equation 1>
           <a series of addresses> <equation 2>
           ..................................
           <a series of addresses> <equation n>  )
```

Each equation defines a substitution to be done: the left-hand side is the old expression which should be replaced by the right-hand side. The addresses specify in which formulae the substitution that follows should be carried out. The form of the addresses can be best explained on examples. If the address is an atomic name of a quantity, e.g. gamma, riemann or rie, then the substitution that follows should be attempted in each component of the quantity. If the name is followed by sets of indices, then the substitution should be attempted only in the components defined by these sets of indices. For example, if the addresses are

```
gamma (0 0 2) (0 0 3) riemann (0 3 0 3) rie L = R
```

then R should be substituted for L in the components ${\Gamma^0}_{02}$ and ${\Gamma^0}_{03}$ of the "gamma", in the tetrad component R_{0303} of the Riemann tensor and in all the coordinate components of the Riemann tensor.

The atomic names can be also followed by symbols defining the valence, the ones described in sec. 16. For some quantities (e.g. for all the tetrad components or for the Christoffel symbols) the valence is fixed forever and so need not be specified. For some others it is relevant. For instance

```
rie (+ + + +) (+ + - -) (0 1 0 1) (0 3 0 3) (- - - -)  ricci
ric (+ -) (0 1) L = R
```

means that R should be substituted for L in all the coordinate components $R^{\alpha\beta\gamma\delta}$ of the Riemann tensor, in the coordinate components ${R^{01}}_{01}$ and ${R^{03}}_{03}$, and in all the coordinate components $R_{\alpha\beta\gamma\delta}$, then in all the tetrad components of the Ricci tensor and in the coordinate component ${R^0}_1$. If an equation is not preceded by any address (i.e. is immediately preceded by another equation or by the atom "substitutions"), then the substitution it defines should be attempted just everywhere. This is rarely reasonable and results in an unnecessary extension of the calculation time.

Note that those substitutions which should be carried out everywhere are attempted first, before the addressed ones are considered. Thus, if some substitutions are addressed and some others are not, the user does not have a full control of the order in which they will be executed.

The following example may show the usefulness of substitutions. If the Einstein field equations are yet to be solved for a metric, then the "ematrix" will contain some unknown functions. After the Einstein tensor is calculated and the field equations are solved, these functions become explicit expressions. For instance, let:

$$ds^2 = f(r)dt^2 - g(r)dr^2 - r^2(dh^2 + \sin^2 h dp^2)$$

.
For this metric, the data are:

```
(coordinates t r h p)
(functions f (r) g (r) )
(ematrix  (f ^ (1 2))   0   0   0
          0   (g ^ (1 2))   0   0
          0   0   r   0
          0   0   0   (r * (sin h))  )
```

From the field equations $G_{ij} = 0$ it follows then that $f = 1/g = 1 - 2m/r$, where m is a constant. In order to check that this is a solution it is enough to add just two more items to the data listed above:

```
(constants m)
(substitutions f = ((1 - 2 * m / r) ^ (1 2))
               g = (1 / (1 - 2 * m / r) ^ (1 2))  )
```

Actually, it would be more reasonable to write

```
(substitutions ematrix f = ((1 - 2 * m / r) ^ (1 2))
               ematrix g = (1 / (1 - 2 * m / r) ^ (1 2)))
```

because after f and g are replaced by their values in the ematrix, they will never appear later and the program need not look for any other opportunity to replace them.

This rather simple example was chosen just for its simplicity. Ortocartan was successfully tested on several much more complicated metrics (see Appendix E).

In general, the user is advised to specify the places in which the substitutions should be carried out as precisely as possible. It makes the calculation faster: the program does not look for opportunities to make the substitutions in the formulae where these substitutions are not necessary. Every unsuccessful attempt at performing a substitution is communicated to the user in the printout. Suppose for instance that the user has written in the substitutions:

```
riemann (0 1 0 1) (0 2 0 2) (0 3 0 3)
     (x ^ 2) = (r ^ 2 - y ^ 2 - z ^ 2)
```

but the variable x did not appear in R_{0202} at all. Then the program will print the following message above R_{0202}:

```
> riemann          DID NOT PROVIDE ANY OPPORTUNITY TO PERFORM
          0 2 0 2

> THE FOLLOWING SUBSTITUTIONS

   2    2    2    2
> X  = R  - Y  - Z
```

This means that in the next run of the program with the same data, the address of this substitution should be riemann (0 1 0 1) (0 3 0 3). This is not an error message, and the program will continue to work.

These messages can be turned off altogether. This happens when one inserts the atom "messages" into the argument (dont print ...) (see sec. 20 and example IV in Appendix E).

Sometimes the whole value of some quantity should be substituted by some other expression. The user may in this case avoid the necessity to rewrite the whole expression by writing the atom "actual" as the left-hand side of the appropriate equation. So for example the equation:

```
riemann (0 1 0 1)  actual = (x ^ 2)
```

means: "from now on the tetrad component R_{0101} of the Riemann tensor will be equal to x^2". The previous value is forgotten.

The equations defining the substitutions are printed in the mathematical format above all the results, after the "symbols", preceded by the heading "substitutions", and with the sub-headings which either say "everywhere" or are the series of addresses as written by the user. One can avoid having the list of substitutions printed (sometimes it is long) by inserting the atom "substitutions" into the argument (dont print ...) (see sec. 20).

If some of the left-hand sides is a sum or a product, then the substitution required may be missed in some cases. One of such cases is when the user wants the sum $x + y$ to be substituted by U, and in the expression now processed there is the sum $x + 2y$. Then the program will not be able to recognize that $x + 2y$ should be replaced by $y + U$. The same concerns a product like ab: the program will not recognize it as being a part of an expression like ab^2. If this difficulty is likely to appear, then it is advised that instead of $x + y = U$ the substitution $x = U - y$ be declared. The same concerns products: instead of $(a * b) = c$ one may write $a = (c/b)$ or $b = (c/a)$. The general principle is: the left-hand sides of the substitutions should be literally present in the expressions where they are to be substituted for.

Another example of a trouble: if a sum is multiplied by a factor, then the substitution for the non-multiplied sum might be missed. For example, if $X+Y = U$, then the program will not recognize that $aX + aY = aU$. However, the facility of substitutions is flexible enough to overcome such difficulties with a little help from the user.

The facility of substitutions may help to bypass some conventions of our system, were they inconvenient in a particular case. For example, the program changes any even power of $\cos x$ to the appropriate power of $(1-\sin^2 x)$ just for uniqueness. Were this inconvenient, one may define an atomic symbol, e.g. COSX, to stand for $(\cos x)$, and then write the equation:

```
COSX = (cos x)
```

in the "symbols", and the equation:

```
(cos x) = COSX
```

in the "substitutions". Then, COSX will appear instead of (cos x) in all the prints, and will not be replaced by (sin x). See example V in Appendix E.

In addition to the literal substitutions described above one can also carry out another kind of substitutions in which the left-hand side of an equation defines a pattern to be looked for, and the right-hand side says what to do with an expression that matches the pattern. For these substitutions one should insert an item of the form:

```
(markers <a series of arbitrary atomic symbols>)
```

into the data. For example:

```
(markers M1 M2 M3 M4)
```

If such a command is found in the data, then the program will understand that M1, M2, M3 and M4 will be used to stand for anything in the substitutions. With the markers, the user can define just a pattern to which an expression should conform in order to be replaced by some other expression. For example, if one writes in the substitutions:

```
((sin x) ^ M1) = ((1 - (CS x) ^ 2) * (sin x) ^ (M1 - 2))
```

then any power of $\sin x$ will be replaced by $(\sin x)^{\text{the old power}-2}(1 - CS^2 x)$, where $CS(x)$ is a user-defined function (note the danger: if $\sin x$ happens to appear with an exponent that is negative or fractional or even is not a number at all, then the substitution will be performed anyway). If one would like such a substitution to be performed not only for $\sin x$, but for sin of any arbitrary argument, then it is enough to use another of the markers instead of x, e.g.:

```
((sin M2) ^ M1) = ((1 - (CS M2) ^ 2) * (sin M2) ^  (M1 - 2))
```

One should not worry about using the same marker in different formulae, each time with a different meaning - the program assigns a meaning to each marker in each substitution anew and does not remember the meanings the markers could have had before.

Suppose, the cartesian radius $r = (x^2 + y^2 + z^2)^{1/2}$ appears in a calculation. One can then place r (x y z) in the "functions" and write:

```
(substitutions (der x r) = (x / r)
          (der y r) = (y / r)
          (der z r) = (z / r)  )
```

With the "markers", the same effect can be achieved by writing just one equation:

```
(substitutions (der M1 r) = (M1 / r))
```

Let us stress that markers can symbolize not only atoms, but any arbitrarily complicated expressions. For instance, with:

```
(substitutions (M1 ^ (1 2)) = (M1 / M1 ^ (1 2)) )
```

also an expression like $(a^2 + b^3)^{1/2}$ will be replaced by $[a^2/(a^2 + b^3)^{1/2} + b^3/(a^2 + b^3)^{1/2}]$. If M1 and M2 are markers, then (M1 + M2) means "any sum", and (M1 * M2) means "any product". For instance, the equation (2 * (log (M1 + M2))) = (log ((M1 + M2) ^ 2)) means: whenever a logarithm of any sum is multiplied by 2, square the sum and drop the coefficient 2. The "markers" can also be used to represent arbitrary parts of sums and products, provided their meaning can be guessed uniquely. For instance, (A + B + M1 + C + M2 + D) means "a sum in which anything stands between B and C and anything stands between C and D", (A + B + M1) means "any sum starting with A + B", but (A + M1 + M2 + C) is not unique: if several terms stand between A and C, then there will be no way to tell which of them should go into M1 and which into M2. Such a pattern will result in an error-message.

The equation:

```
M = (M * a * b ^ 2 / c)
```

with M being a marker, means "multiply the equation, whatever it was, by ab^2/c". In this case, M stands for the whole old expression. This is sometimes useful, but the address(-es) of this substitution must be specified, or else **all** equations will be multiplied by the factor ab^2/c.

The disadvantage of the substitutions by pattern-matching is that they are slower than the literal substitutions, and sometimes can be very much slower (when the right-hand sides are complicated functions of the "markers").

Note that Ortocartan understands the functions "quotient" and "remainder" acting on the "markers". However, it is the user's exclusive responsibility to make sure that the arguments of these functions will in each case turn out to be integer numbers.

Expanding integer powers of sums is a part of the substitutions. For instance, in order to expand $(A + B)^4$ one should write in the substitutions:

```
((A + B) ^ 4) = expand
```

Should just any power of $(A + B)$ be expanded, one writes:

```
((A + B) ^ M1) = expand
```

where M1 should be one of the markers.

The user writes the substitutions by pattern-matching together with the literal ones in the argument (substitutions.....), without marking them in any way. The presence of the argument (markers.....) is a sufficient warning for Ortocartan - it will then be able to distinguish the two kinds of substitutions by itself.

19 Substitutions in the data.

This facility was introduced long ago, before easy-to-use text-editors became commonly available, and before the additional programs of the Appendices B and C were written. It is unlikely that any user would find this useful today. However, the facility still exists, so its description is retained.

As mentioned before, the data written by the user are, before applying, processed by the procedure of algebraic simplification, so that the user may write them in a form that is more compact than the final one (e.g. one may leave the task of carrying out a multiplication by a sum, or calculating a derivative, to the program itself). For this reason, however, the final form of the data may appear inconvenient (e.g. clumsy or too extended). In such a case the result may be smoothed out by the user, either by rewriting the input data, or by applying the substitutions. Consider, however, the following example (taken from a paper by J. Kowalczyński). Let:

$$S \stackrel{\text{def}}{=} [2m + 2b\log(ZC - ap) + b\log(q)]a/p - 4bZC/p^2 - (1/3)hp^2, \tag{1}$$

where a = 1 or a = -1, b, c, h and m are constants, t, ϕ, r and z are coordinates and

$$ZC = z + c, \qquad p = (t^2 - r^2)^{1/2}, \qquad q = p^8/(t - r\cos(\phi))^8.$$

The symbol S appears in the metric and so is differentiated during the calculation. However, the automatically calculated derivatives of S are rather messy, for example:

```
                -2  2    2 -1/2                -2  2    2 -1/2
dS/dt = - abtp  (t  - r )     log(q) - 2abtp  (t  - r )     log(ZC-ap)

          6 -1                 -8   2    2 -1/2
   + 8abtp q  (t - r cos(phi))   (t  - r )

         7 -1                -9        -2  2   2 -1/2
   - 8abp q  (t - r cos(phi))   - 2amtp  (t - r )

         -3     2    2 -1/2              2    2 -1/2
   + 8btp  ZC (t  - r )     - (2/3)htp (t  - r )

       2   -1         -1  2    2 -1/2
   - 2a btp  (ZC - ap)  (t  - r )    .                                    (2)
```

The missing simplifications are:

```
 2     2    2
t   = r  + p ,                                                           (3)
```

```
 -1                   8  8
q   = (t - r cos(phi)) /p ,                                          (4)

                                  -1  3     -1          -1       -1
log(q) = - 2 log(ZC -ap) + (1/3)ab  hp  + ab  Sp  + 4ap  ZC - 2b  m,  (5)

 2
a  = 1.                                                              (6)
```

After they are done, the more neat result is obtained:

```
             -3       -1                -1       -4
dS/dt = 8abtp   - 8abp  (t - r cos(phi))   + 4btp  ZC

           -2         -1           -2
     - 2btp  (ZC - ap)   - ht - tSp  .                               (7)
```

The net result of having the substitution (7) carried out may be effected by declaring the equation (1) in the "symbols", and then writing the equations (3), (4), (5) and (6), in turn, in the "substitutions". Then, however, such a coupled series of substitutions would be performed a lot of times, each time inside a large expression, and this would make the calculation slow. It is more economical to force the program to replace every occurrence of dS/dt directly by (7).

This can be done. One should declare S as one of the "functions", and then write:

```
(substitutions
        (der t S) = (deriv t (<the right-hand side of (1)>)))
(data substitutions
        (der t S) (t ^ 2)  = (r ^ 2  +  p ^ 2)
        (der t S) (q ^ -1) = ((t - r * (cos phi)) ^ 8 / p ^ 8)
        (der t S) (log q) = (-2 * (log(ZC  -  a * p)) +
                                        (1  3) * a * h * p ^ 3 / b
                      + a * S * p / b + 4 * a * ZC / p  -  2 * m / b)
        (der t S) (a ^ 2) = 1 )
```

(For brevity we omit the corresponding substitutions in dS/dϕ, dS/dr and dS/dz).

The "data substitutions" specify the replacements to be made inside the "substitutions" before the program starts to run. They have the same syntax as the substitutions themselves, with the difference that here the addresses are the appropriate left-hand sides of the equations from the "substitutions", and the replacements are performed in the corresponding right-hand sides. If no address is given, then the appropriate data substitution is performed just everywhere in the "substitutions".

In the example given above the value of dS/dt, initially equal to the right-hand side of (2), after being in turn transformed by (3), (4), (5), and (6) will finally change into the right-hand side of (7), and then just this expression will be directly substituted for dS/dt in the calculation proper.

The use of the data substitutions facility has the following result:

1. The substitutions are standardized and printed as explained in section 19, just as if there were no "data substitutions".

2. The data substitutions are printed analogously, immediately below, with the heading:

(THE SUBSTITUTIONS LISTED ABOVE WILL BE THEMSELVES TRANSFORMED BY THE FOLLOWING SUBSTITUTIONS)

(Don't get confused if you had turned off printing the list of substitutions - then they will not be "listed above" although the heading of data substitutions will say so). Each equation printed has its own sub-heading which says either EVERYWHERE, or:

```
> IN THE VALUE OF THE EXPRESSION <the appropriate  left-hand
         side from substitutions>
```

3. Another heading is printed:

(THE SUBSTITUTIONS YOU ASKED ME TO MODIFY WILL HAVE THE FOLLOWING FINAL FORM)

4. Those substitutions which were modified are printed in their final, working form.

One can avoid having the list of modified substitutions printed by inserting the atom "modifications" into the argument (dont print ...) (see sec. 20).

20 Suppressing some parts of the printout.

Sometimes the user is not interested in having all the results of the routine run of the program. In this case he or she may instruct the program not to print some intermediate results or to stop its work earlier than usually, before calculating the Weyl tensor. This results in saving paper as well as computer's time and core, and, consequently, user's time, too.

The request to stop the work earlier is expressed by inserting into the data the item of the form:

```
(stop after <name>)
```

where `<name>` is one of the names from the dictionary in section 8 (only scalars and tetrad-names will have the required result). For example, if the expression:

```
(stop after riemann)
```

is found in the data, then the Ricci and Weyl tensors will not be calculated.

If the user is not interested in some intermediate results, then the appropriate request is:

```
(dont print <a series of names>)
```

where `<a series of names>` consists of any number of names from the dictionary in section 8 (again only scalars and tetrad names except einstein, i.e. only the quantities that are calculated in the routine run, will have the required result). All the quantities that bear these names, though calculated for the needs of further calculation, will not appear in print. For example, if the request:

```
(dont print ie determinant agamma gamma ricci)
```

is found in the data, then none of the quantities $e^{\alpha}{}_{i}$, $\det(e^{i}{}_{\alpha})$, $\Gamma^{i}{}_{[jk]}$, $\Gamma^{i}{}_{jk}$ and R_{ij} will be printed. The series of names may also contain any of the atoms "substitutions", "modifications", "messages" and "timemessages". Inserting them into the series will have, respectively, the following results: the list of substitutions will not be printed, the list of substitutions modified by data substitutions will not be printed, the messages about the unsuccessful attempts at substitutions will not be printed and the information about timings shown in sec. 9 will not be printed. The order of the atoms in the series is irrelevant.

21 Formatting the output.

The linelength of the output on the screen is 72 characters. However, the user may wish to produce a narrower output (e. g. in order to be printed and inserted in a typed manuscript) or a wider output (e. g. in order to use the paper more economically). It is then possible to change the linelength of the output by inserting the following item into the data for Ortocartan:

```
(rmargin <n+1>)
```

where n is the required new linelength. This will be useful mainly if the user wishes to print the results on wide paper with continuous feeding; in this case the typical parameters are (rmargin 120) or (rmargin 132).

Similarly, you can adjust the left margin by inserting the item

```
(lmargin <n+1>)
```

into the data. This will have an effect only in those formulae that extend over more than one line: the second and following lines will be printed starting at the (n + 1)-st column. The default value for lmargin is 8. The first line of each formula always begins at column 5, and this is not adjustable.

22 Storing the results on a disk file and printing them.

In a normal run, Ortocartan will show the results of the calculation only on the screen. When the calculation is simple (such as the Schwarzschild or the Robertson-Walker metric), the formulae will flash through the screen much faster than you can read them. You can go back and forth along the results to see the details by pressing the buttons "PgUp" and "PgDn" on the keyboard. However, it is often convenient to print the results on paper. In order to do so, you have to click on the "File" in the upper left corner of the screen, and then click the "To file" option in the dropdown menu. Then you can choose the name and the path of the directory where the complete output from Lisp will be stored. Later, you can view it with any text-editor, edit and print it.

23 Special forms of the output.

Ortocartan produces formulae that are easily readable. However, sometimes these formulae have to be used as input for another calculation or inserted in a text for publication. Retyping is always annoying, and carries the risk of typos. This can be largely avoided thanks to the following two conveniences.

If the formulae produced by Ortocartan should be used as input for another run of Ortocartan, then you should insert the following item into the data:

```
(output for input)
```

Then, the whole output will be written in the same notation as is used on input, and every piece of it will be directly readable for Ortocartan as its data. Usually, you will want to use only parts of this output, but then all you have to do is very simple editing (cutting and pasting), without any rewriting.

If the (some of the) formulae produced by Ortocartan should be used as a part of a text for publication, then you should insert the following item into the data:

```
(output for latex)
```

Then, all the formulae will be written as expressions readable for the system Latex. You will only have to add your favourite Latex preamble, and run Latex on them, they will then show up as a neat Latex output, with all equations numbered. Ortocartan will automatically replace the names of the Greek letters with their Latex codes, e.g. if you use "alpha" as the name of the variable, then it will be replaced by `\alpha` in the Latex code, and appear as α in the output from Latex.

The printing in Latex format is not 100% safe against Latex errors. It may happen that the end of line in the output from Ortocartan will occur in a place that is not acceptable for Latex. Then Latex will signal an error. You may also prefer to have the Latex linebreaks in other places than Ortocartan will place them. Such failures will have to be corrected by hand.

24 Errors.

The program Ortocartan has some built-in tests for correctness of the data. If one of our tests finds an error, then an error-message is printed which should directly identify the kind of error that was made. However, only some of the possible errors can be identified uniquely.

For beginners in Lisp computing, the most likely error is incorrect placement of parentheses. If Ortocartan refuses to start working, then almost surely some left parentheses in your data are unmatched. If you have written the data directly from the keyboard, you may try your luck by writing a few right parentheses and pressing "RETURN" again. However, if the parentheses were distributed incorrectly, then a fatal error of one kind or another is sure to occur, and the error-message can be quite beside the point, especially when the data are read from a disk. There is no way for the Lisp system to recognize incorrect distribution of parentheses. It knows what you want it to do only by reading your data as lists, and the parentheses divide the data into sublists. Placing the parentheses incorrectly may turn some sublists into a mathematical nonsense, but they will still be meaningful S-expressions, and Lisp will take them literally. Hence, whenever you get an obvious nonsense as a result, or an overtly absurd or unintelligible error-message, please check carefully the positions of all parentheses in your data.

There is one more possible error that is not signalized at all, and may cause trouble. Each item of the data described in one of the sections 11 to 21 may be used only once. This is not really a limitation of the power of the program, it is just a small piece of rigour forced upon the user, as each item can hold many requests of its appropriate kind. If any of the items is used twice or more times, then only its last appearance has the expected results, and the previous ones are simply ignored. For example, in the call:

```
(ortocartan '(
 (SPHERICAL METRIC IN STANDARD FORM)
 (functions  MU(T R)  NU(T R))
 (coordinates T R THETA PHI)
 (ematrix  (exp NU)  0  0  0  0  (exp MU) 0  0  0  0  R
            0  0  0  0  (r *(sin theta)))
 (tensors riemann  (+ - - -) ricci)
 (stop after ricci)
 (tensors riemann  (- - - -))
 (stop after weyl)
                  ))
```

only the components $R_{\alpha\beta\gamma\delta}$ of the Riemann tensor will be calculated and printed (second appearance of (tensors....)) while the requests to calculate $R^{\alpha}{}_{\beta\gamma\delta}$ and $R_{\alpha\beta}$ will be ignored. Also, the program will finish its work after calculating the Weyl tensor, i.e. after completing the whole routine run, while the request to stop after the Ricci tensor will be ignored.

We recall what has already been said before: if you wish to use capitals and lower case letters as different symbols, then you should insert the command:

```
(setq !*lower nil)
```

before the "(ortocartan'(" line. Without it, all capitals will be mapped into lower case letters and printed as such. Even worse things can occur if you do insert the (setq !*lower nil) command, and then carelessly use capitals and lower case letters. With this command inserted, all function names should be written in lower case, for example sin, but not SIN (the SIN would then be an unknown function for Ortocartan). You can then use capitals only as names of the symbols introduced by yourself in the data for Ortocartan, and in the title.

If the "(setq !*lower nil)" has been used, then it is advisable to place the command

```
(setq !*lower t)
```

at the end of the data for Ortocartan (i.e. behind the two right parentheses). Then Lisp will revert to its default mode after it has finished the calculation, otherwise it will continue to see capitals and lower case letters as different.

ACKNOWLEDGEMENTS

This manual was rewritten into a computer file for its third edition thanks to the courtesy of Professor F. Hehl. We express our gratitude to him and to his collaborators at the University of Cologne who actually did the hard work of retyping. Our thanks are due to Drs J. Richer and A. Norman who reworked Ortocartan into Cambridge Lisp and implemented it on an IBM computer. This enabled one of the authors (A. K .) to rewrite Ortocartan into Slisp/360 (now a defunct version). The kind assistance of Professor J. Fitch in this last task is hereby gratefully acknowledged. The program Calculate (see Appendix B) and two other, now defunct, programs were written and implemented as a part of a project supported by the A. von Humboldt Foundation at the Max Planck Institute in Garching (Germany). The Slisp/360 version of Ortocartan was implemented as a part of a project supported by the Deutsche Forschungsgemeinschaft at the Konstanz University (Germany). The pattern-matching substitutions were added together with several other improvements as a part of a project supported by the Deutsche Forschungsgemeinschaft at the University of Cologne. A. K. is grateful to Professors J. Ehlers, H. Dehnen and F. Hehl who were the respective hosts of those projects for their kind hospitality.

The Atari computer, on which a newer Cambridge Lisp version of Ortocartan was implemented, was bought for funds kindly provided by Professor M. Demiański from his grant. The implementation of Ortocartan on it was facilitated by the assistance of Professor M. A. H. MacCallum. The programs described in Appendix C were first written and implemented with that version.

The most recent implementation in the Codemist Standard Lisp to be run under the Windows 98 and Linux operating systems was made possible thanks to the grant no 2 P03B 060 17 awarded by the Polish Research Committee to a research group at the Institute of

Theoretical Physics led by Professor Andrzej Trautman. The funds from the grant were used to upgrade A.K.'s computer at the N. Copernicus Center and to buy a license for the Codemist Standard Lisp from Codemist Limited. I (A.K.) am very grateful to Professor Trautman for including me in his research group.

A How to acquire Ortocartan.

In order to use Ortocartan one must first buy the Codemist Standard Lisp. It can be bought from:

Professor John Fitch, Director
CODEMIST Limited
"Alta", Horsecombe Vale
Combe Down
BATH, Avon, BA2 5QR
England
phone and fax (44-1225) 837 430
email jpffitch@maths.bath.ac.uk

Ortocartan is free of charge. If you wish to have it, just contact A. Krasiński, I will send you a diskette. How to install Ortocartan when Lisp is already working is described in sec. 7.

B How to use the program Calculate.

The definition of the function "calculate" is contained in the file "calcsl.lis" in the distribution diskette of Ortocartan (see sec. 7). In order to use the program you simply have to follow the instructions from sec. 7. To make sure that the program is indeed at your disposal, after starting Ortocartan, write:

infocalculate

and press "RETURN". The response should be reassuring.

With the exceptions of integration and factoring polynomials, this program can carry out any kind of elementary algebraic operations, but lacks the facility of writing programs in it without resorting to Lisp, i.e. is not a programming language in itself.

The format for the user's data is nearly identical to the one described in the manual (see example V in Appendix E). The following differences with respect to the description in the manual must be observed here:

Section 1:

The program can simplify symbolic expressions of (in principle) arbitrary degree of complexity. In fact, it gives the user direct access to the operations which are carried out as intermediate tasks in the program Ortocartan.

Section 2:

Does not apply here at all.

Section 7:
Whenever you start Ortocartan, the function "calculate" is ready for your use, too.

Section 8:
Does not apply here at all.

Section 9:
Several expressions can be simplified in one run of the program. First, the heading is printed:

I UNDERSTAND YOU REQUEST THE FOLLOWING EXPRESSION TO BE SIMPLIFIED

Then, the program prints the expression as written by the user on input translated from the input format of sec. 5 to the normal mathematical format (without yet simplifying it). Next, the program prints:

THE RESULT IS

and the simplified expression follows. The simplified expression is printed as the equation

```
result    = <the expression>
       <i>
```

where <i> is a subscript running consecutively from 1 for the first expression. Such labeling of the printed results makes it possible to direct the substitutions to different results.

No time-messages are printed for single operations, only the total run-time for the call to calculate will be displayed.

Section 10:
The first line here should be:

(calculate '(

The second item of the data is again a title, the last line must contain the two right parentheses.

Section 11:
The number of coordinates (i.e. independent variables in differentiation) is arbitrary here. Coordinates can be omitted. If no differentiations are going to be done, then all the symbols used can be called constants.

Section 14:
The heart of the data are here the expressions to be simplified. They are written as:

```
(operation <an arbitrary expression written according to the rules of
             section 5>)
(operation <another arbitrary expression>)
.......................................
(operation <another arbitrary expression>)
```

The number of operations is arbitrary. It is understood that the program will be used to calculate complicated derivatives or to multiply large polynomials or to carry out series of substitutions in given expressions.

Section 16:
Does not apply here at all.

Section 18:
The substitutions should be directed here to `result (<index>)` where the `<index>` is the subscript that appears in print. If no address is specified, then they will be carried out everywhere, i. e. in each result in the present call to calculate. All the rules explained in sec. 18 apply here without any modification.

Section 20:
Does not apply here at all.

C The programs "ellisevol", "curvature", "landlagr", "eulagr" and "squint".

C.1 The program Ellisevol.

This program calculates all the quantities appearing in the evolution equations of the kinematical tensors of fluid flow, as defined by Ellis[2]. Since all these equations are consequences of the Ricci identity $u_{\alpha;\beta\gamma} - u_{\alpha;\gamma\beta} = u_\mu R^\mu{}_{\beta\gamma\delta}$, they will be fulfilled identically in many cases. However, they may fail to be identically fulfilled when we make assumptions about separate parts of the flow, e.g. if we assume that the shear is zero. As is well known, such assumptions have consequences for the other characteristics of the flow, and the Ellis equations will show what the consequences are. Along the way, the program calculates the expansion, acceleration, the shear tensor and scalar, the rotation tensor and scalar, and the electric and magnetic parts of the Weyl tensor with respect to the velocity vector.

In order to make sure that the program Ellisevol is indeed contained in the core-image of Lisp that you use, write:

infoellis

and press "RETURN".

Since the signature assumed in the calculation is $(+ - --)$, the formulae may differ from those given in textbooks, and so some of them are quoted below for reference.

The program is called by writing:

```
(ellisevol'(
```

[2]G. F. R. Ellis, in *General relativity and cosmology. Proceedings of the International School of Physics "Enrico Fermi", Course 47: General Relativity and Cosmology.* Edited by R. K. Sachs, Academic Press, New York 1971.

in the first line of the data file. All the remaining parts of the data are the same as for the program Ortocartan described in the manual, except that there is one additional item here:

```
(velocity <contravariant tetrad components of the velocity field
u0, u1, u2, u3>)
```

(see Example VII in Appendix E). This is the vector field for which all the kinematical quantities will be calculated. This argument cannot be omitted, the omission would result in an error, communicated by the program Ellisevol.

Note: It is implicitly assumed that this will be a velocity field of some continuous medium because this is the most frequent application of these formulae. However, the calculation makes sense for any timelike vector field whose length is normalized to 1. In particular, this may the the unit vector collinear with a timelike Killing vector.

The second item of the data must be the title (this rule applies to all the additional programs described in this appendix).

The quantities printed are the "ematrix", the velocity field (tetrad components as given by the user), the determinant of the ematrix, the inverse matrix to the "ematrix", the contravariant coordinate components of the velocity field (named "uvelo"), the covariant coordinate components of the velocity field (named "lvelo"), the components of the metric tensor (named "metric"), the inverse metric (named "invmetric"), the "agamma" and the "gamma", and the Christoffel symbols (named "christoffel"). Then come:

The matrix of covariant derivatives of the covariant velocity field (named "vtida" – for tidal matrix of velocity).

The contravariant ("uacce") and covariant ("lacce") components of the acceleration field.

The rotation tensor $\omega_{\alpha\beta}$, named "rotdd".

The mixed rotation tensor $\omega_\alpha{}^\beta$, named "rotdu".

The square of the rotation scalar.

The covariant ("projdd") and mixed ("projdu") components of the projection tensors $g_{\alpha\beta} - u_\alpha u_\beta$ and $\delta^\alpha{}_\beta - u^\alpha u_\beta$.

The expansion scalar.

The covariant ("sheardd") and mixed ("sheardu") components of the shear tensor, and the square of the shear scalar.

The matrix of covariant derivatives of the covariant acceleration, $\dot{u}_{\alpha;\beta}$ (called "atida" for the tidal matrix of the acceleration).

The rotation constraint equations:

$$\omega_{[\alpha\beta;\gamma]} + \dot{u}_{[\alpha;\gamma}u_{\beta]} + \dot{u}_{[\alpha}\omega_{\beta\gamma]} = 0,$$

(square brackets on indices denote antisymmetrization, round brackets on indices denote symmetrization). The components of these equations are printed with the name "rotcons".

The shear constraint equations:

$$h^{\alpha}{}_{\beta}(\omega^{\beta\gamma}{}_{;\gamma} - \sigma^{\beta\gamma}{}_{;\gamma} + \frac{2}{3}\theta^{;\beta}) - (\omega^{\alpha}{}_{\beta} + \sigma^{\alpha}{}_{\beta})\dot{u}^{\beta} = 0$$

under the name "shearcons".

The rotation evolution equations:

$$h_{\alpha}{}^{\gamma}h_{\beta}{}^{\delta}\dot{\omega}_{\gamma\delta} - h_{\alpha}{}^{\gamma}h_{\beta}{}^{\delta}\dot{u}_{[\gamma;\delta]} + 2\sigma_{\delta[\alpha}\omega^{\delta}{}_{\beta]} + \frac{2}{3}\theta\omega_{\alpha\beta} = 0,$$

under the name "rotevol".

The tetrad components of the Riemann and Ricci tensors, and the curvature scalar.

The Raychaudhuri equation:

$$\dot{\theta} + \frac{1}{3}\theta^2 - \dot{u}^{\alpha}_{;\alpha} + \sigma^{\alpha\beta}\sigma_{\alpha\beta} - \omega^{\alpha\beta}\omega_{\alpha\beta} + R_{\alpha\beta}u^{\alpha}u^{\beta} = 0$$

under the name "raychaudhuri equation".

The tetrad components of the Weyl tensor.

The (coordinate) electric components of the Weyl tensor:

$$E_{\alpha\beta} = C_{\alpha\rho\beta\sigma}u^{\rho}u^{\sigma} \equiv E_{\beta\alpha}$$

under the name "elweyl".

The shear evolution equations:

$$h_{\alpha}{}^{\gamma}h_{\beta}{}^{\delta}\dot{\sigma}_{\gamma\delta} - h_{\alpha}{}^{\gamma}h_{\beta}{}^{\delta}\dot{u}_{(\gamma;\delta)} + \dot{u}_{\alpha}\dot{u}_{\beta} + \omega_{\alpha\gamma}\omega^{\gamma}{}_{\beta} + \sigma_{\alpha\gamma}\sigma^{\gamma}{}_{\beta} + \frac{2}{3}\theta\sigma_{\alpha\beta}$$

$$+\frac{1}{3}h_{\alpha\beta}[2(\omega^2 - \sigma^2) + \dot{u}^{\gamma}_{;\gamma}] + E_{\alpha\beta} = 0,$$

under the name "shearevol".

The magnetic components of the Weyl tensor:

$$H_{\alpha\beta} = \frac{1}{2}\sqrt{-g}\varepsilon_{\alpha\gamma\mu\nu}C^{\mu\nu}{}_{\beta\delta}u^{\gamma}u^{\delta} \equiv H_{\beta\alpha}$$

under the name "magweyl", the $\varepsilon_{\alpha\gamma\mu\nu}$ is the Levi-Civita symbol.

The "magnetic constraint" equations:

$$2\dot{u}_{(\alpha}w_{\beta)} - \sqrt{-g}h_{\alpha}{}^{\gamma}h_{\beta}{}^{\delta}(\omega_{(\gamma}{}^{\mu;\nu} + \sigma_{(\gamma}{}^{\mu;\nu})\varepsilon_{\delta)\rho\mu\nu}u^{\rho} = H_{\alpha\beta},$$

under the name "magcons", where w_{β} is the rotation vector field defined by:

$$w^{\alpha} = \frac{1}{2\sqrt{-g}}\varepsilon^{\alpha\beta\gamma\delta}u_{\beta}\omega_{\gamma\delta}.$$

Each of the names of the quantities printed can be used as an address in the substitutions and as an entry in the (dont print ...) or (stop after ...), just like in the main program Ortocartan. For the print-names that consist of two words (like the "raychaudhuri equation"), only the first word should be used as an address or as an entry in (dont print ...) and (stop after ...).

All the facilities described for Ortocartan exist also here, including the "tensors" – except that the Christoffel symbols and the metric are calculated in every run because they are needed at later stages of the calculation (but their printing can be suppressed with (dont print ...)).

Please do not forget about the last line with the two right parentheses. This applies to all the other programs of this appendix.

C.2 The program "curvature".

This program calculates the curvature tensor from given connection coefficients in any number of dimensions (see example VIII in Appendix E). The connection coefficients are assumed symmetric (i.e. torsion-free), but need not be metrical. The program was written for one special application, and hence the assumption of zero torsion; it can be removed without any difficulty.

To make sure that this program is in your core-image, write:

infoncurva

and press "RETURN".

The first item of the data for this program is the line:

```
(curvature <n> '(
```

where `<n>` is the number of dimensions of the manifold on which the connection coefficients and the curvature are defined. The next item should be the title of the problem, just like in every other program of the Ortocartan collection. The main part of the data is the specification of the connection coefficients that has the form:

```
(connection <list of components of the connection coefficients
in the order D000 D001 D002 D003 ... D00n D011 D012 ... D01n ...
Dn00 Dn01 ... Dnnn>)
```

i.e. only the algebraically independent components of $D^i{}_{jk}$ with $j \leq k$ are given.

All the other parts of input, like constants, functions, symbols, substitutions, etc, can be used just like in Ortocartan, if only they make sense here (for example (stop after ...) or (dont print ...) would not work here because there is only one quantity that is being calculated).

The last item of the data are the two right parentheses.

C.3 The program "landlagr".

This program calculates the reduced lagrangian for the Einstein equations by the Landau–Lifshitz method[3]. This is the Hilbert lagrangian with the divergences containing second derivatives of the metric already removed. In short, this (noncovariant) lagrangian is obtained by deleting from the scalar curvature those terms in which the Christoffel symbols are differentiated, and taking the remaining part with the reverse sign.

To make sure that this program is in your core-image, write:

infolandlagr

and press "RETURN".

The program is called by writing at the beginning of the list of data:

```
(landlagr'(
```

The remaining data are exactly like for "ortocartan".

The main part of the data is the "ematrix", and the quantities printed are the ematrix and its inverse, the metric and its inverse, the "agamma", the "gamma", the Crhistoffel symbols and the lagrangian. You can direct substitutions to it with the address "lagrangian". See example IX in Appendix E.

The last item of the data are the two right parentheses.

Users are warned that deriving the Einstein equations from a variational principle with assumptions limiting the generality of the metric is tricky and requires detailed knowledge about the problem. It may happen that the Euler-Lagrange equations will have nothing to do with the Einstein equations; this is known, for example, to occur for certain Bianchi-type models[4]. Therefore, the user must make sure, before using the program "landlagr", that the situation is appropriate for using the lagrangian methods.

C.4 The program "eulagr".

This program calculates the Euler-Lagrange equations starting from a lagrangian specified by the user. It is assumed that the resulting E-L equations will be ordinary differential equations (i.e. that there is only one independent variable in the lagrangian action integral).

To make sure that this program is in your core-image, write:

infoeulagr

and press "RETURN".

The program is called by writing as the first item of the data:

[3]L. D. Landau and E. M. Lifshitz, Teoria polya, 6th Russian edition. Izdatelstvo "Nauka", Moskva 1973, sec, 93.

[4]See M. A. H. MacCallum, in *General relativity. An Einstein centenary survey.* Edited by S. W. Hawking and W. Israel. Cambridge University Press 1979, p. 552.

```
(eulagr'(
```

The second item of the data is the title.

The main part of the data are the following three items (see example X in Appendix E):

```
 (parameter <an atomic name of the independent variable>)
 (variables <the list of names of the lagrangian variables,
             an arbitrary number of them>)
 (lagrangian <the formula for the lagrangian written in accordance
              with the rules of Section 5>)
```

The "variables" must be declared as functions in the (functions ...) list because the program allows them to depend also on other arguments in addition to the "parameter" specified in the data. (For example, you may wish to differentiate some of the E-L equations with respect to another variable.)

The consecutive Euler-Lagrange equations are printed with the headings:

```
THIS IS THE VARIATIONAL DERIVATIVE BY <the name of the lagrangian variable>
```

as the equations

```
eulagr        = <the appropriate equation>
      <index>
```

The names and indices are needed to address the substitutions. Whatever devices of the main program Ortocartan make sense here, can be used, and all of Ortocartan's conventions apply. If you wish to stop the calculation at an earlier stage, you can write:

```
(stop after lagrangian)
```

(this would make sense if you want to just check whether the lagrangian you wrote in the data has no errors in it), or:

```
(stop after (eulagr <n>))
```

where `<n>` is the index of the last Euler-Lagrange equation that you want to have. Then, the whole expression `eulagr <n>` has to be in parentheses.

The last item of the data are the two right parentheses.

C.5 The program "squint".

This program verifies whether a given expression is a first integral of a given set of ordinary differential equations. The program was written for a specific application, therefore it is rather limited in its abilities. It is assumed that the (hypothetical or actual) first integral is a polynomial of first or second degree in the first derivatives of the functions that should obey the set of equations. It is also assumed that the equations in the set are all of second or first order.

To make sure that this program is in your core-image, write:

infosquint

and press "RETURN".

The program is called by writing as the first item of the data:

```
(squint'(
```

(This is an abbreviation for "square integral".)

The second item of the data is the title.

The main part of the data are the following three items (see example XI in Appendix E):

```
(parameter <an atomic name of the independent variable
           in the set of equations>)
(variables <the list of names of the functions that should obey
           the set of equations>)
(integral <the formula for the first integral
           to be tested by the program,
           written in accordance with the rules of Section 5>)
```

The "variables" must be declared as functions in the (functions ...) list because the program allows them to depend also on other arguments in addition to the "parameter" specified in the data. The "integral" may either be an explicit expression which is being tested by the program whether it is a first integral indeed, or a polynomial of second or first degree in the first derivatives of the "variables" with unknown coefficients. The unknown coefficients must then be declared as (functions ...) of the appropriate variables.

Example[5]: suppose you have a set of 3 second-order ordinary differential equations to be fulfilled by the functions $f^1(t)$, $f^2(t)$ and $f^3(t)$, of the form:

$$\frac{d^2 f^i}{dt^2} = W^i{}_{jk} \frac{df^i}{dt} \frac{df^j}{dt} + V^i{}_j \frac{df^j}{dt} + U^i \qquad (C.1)$$

[5]For an actual example of a simple application of this program, see Example XI in Appendix E.

(the coefficients ${W^i}_{jk}$, ${V^i}_j$ and U^i are functions, but their exact forms are irrelevant here). Suppose you expect the following expression to be a first integral of this set:

$$I = Q_{ij}\frac{df^i}{dt}\frac{df^j}{dt} + L_i\frac{df^i}{dt} + E,$$

where Q_{ij}, L_i and E are functions (as yet unknown) of the variables f^i. Then the "integral" in the data for "squint" should be:

```
(integral (Q11 * (der t f1) ^ 2 + 2 * Q12 * (der t f1) * (der t f2)
    + 2 * Q13 * (der t f1) * (der t f3) + Q22 * (der t f2) ^ 2
    + 2 * Q23 * (der t f2) * (der t f3) + Q33 * (der t f3) ^ 2
    + L1 * (der t f1) + L2 * (der t f2) + L3 * (der t f3) + E) )
```

In this case, the other parts of the main data should be:

```
 (parameter t)
 (variables f1 f2 f3)
 (functions f1(t) f2(t) f3(t) Q11(f1 f2 f3) Q12(f1 f2 f3) Q13(f1 f2 f3)
            Q22(f1 f2 f3) Q23(f1 f2 f3) Q33(f1 f2 f3) L1(f1 f2 f3)
            L2(f1 f2 f3) L3(f1 f2 f3) E(f1 f2 f3) )
```

(you may allow all the "functions" to depend on more variables if you wish, for example when you suspect that the coefficients of the first integral will explicitly depend also on t.) The number of the "variables", and consequently of the equations in the set, is arbitrary.

The program will then print the resulting partial differential equations to be fulfilled by Q_{ij}, L_i and E (but it is up to you then to solve them).

First, the program calculates and prints the total derivative dI/dt in the form of the equation:

```
maineq = <the calculated derivative dI/dt>
```

You can prevent this expression from being printed (it will likely be quite long) by writing:

```
(dont print maineq)
```

However, it is not reasonable to allow the program to calculate dI/dt without taking into account the set of equations (C.1). Hence, you should insert the following item into the data:

```
(substitutions
   maineq (der t t f1) = <the right-hand side from (C.1)>
   maineq (der t t f2) = <the right-hand side from (C.1)>
   maineq (der t t f3) = <the right-hand side from (C.1)>
          )
```

In this way, the program will eliminate second derivatives of the f^i, thus making use of the set (C.1). The "maineq" means that these three substitutions should be carried out within the dI/dt (the "main equation"), before the program goes on with the calculation (see below).

Next, the program finds and prints the coefficients of all the expressions $\frac{df^i}{dt}\frac{df^j}{dt}\frac{df^k}{dt}$, $\frac{df^i}{dt}\frac{df^j}{dt}$, $\frac{df^i}{dt}$ and the terms that do not contain any derivatives of the f^i. These coefficients are printed with the headings that have the following form:

```
                                      2
>   THIS IS THE COEFFICIENT OF  f2,    f3,
                                   t      t
```

and the coefficients themselves are printed in the following form:

```
>   equation  = <the appropriate expression>
            6
```

For a set of n equations to determine n functions, there will be $\frac{1}{6}(n+1)(n+2)(n+3)$ such coefficients altogether. If you do not need some of them, you can insert the following item into the data:

```
(dont print equation (<n1>) (<n2>) (<n3>) ...)
```

where the `<n1>, <n2> <n3>` are any numbers. (Note: just like in the program Ortocartan, each of these numbers should be put in parentheses.) If you wish the program to stop working before it has calculated all the coefficients, write the following item in the data:

```
(stop after (equation (<n>)) )
```

where `<n>` is the number of the last equation to be printed. Note again: here the `equation <n>` must be put in parentheses, and the number `<n>` must be in its own parentheses. This may be confusing, but has a justification in the algorithm of the program. Since the justification would be highly technical, we shall skip it. You can also write:

```
(stop after maineq)
```

and then the program will not print any of the coefficients.

In practice, you will solve the equations one by one, and substitute the solutions in the remaining equations until all of them are satisfied. When this finally happens, the program does not print either the "maineq" or the "equations", but just prints the following message:

```
THE FIRST INTEGRAL IS ALREADY MAXIMALLY SIMPLIFIED
AND IS EXPLICITLY CONSTANT
```

The substitutions are carried out when you write more equations in the (substitutions ...) list shown above. You can direct the additional substitutions either to "maineq" or to any "equation". Directing the additional substitutions to "maineq" is in most cases not reasonable. This is often a long and complicated expression, and the substitution may take quite a while. It is more reasonable to direct those additional substitutions to the "equations". This is done by writing:

```
equation (<n1>) (<n2>) (<n3>) .....
```

in front of the appropriate substitution, where the numbers `<n1>, <n2>, <n3> ...` (each necessarily in parentheses) refer to the equation numbers.

Whatever devices of the main program Ortocartan make sense here, can be used.

The last item of the data are the two right parentheses.

The "integral" need not necessarily be a polynomial of second degree in the $f^i{}_{,t}$. It can be a polynomial of first degree in $f^i{}_{,t}$, or it may be independent of the derivatives. However, the program "squint" will go wild and produce a nonsense result when you try an "integral" that is a polynomial in $f^i{}_{,t}$ of any degree higher than 2 or if the second derivatives of the f^i are not eliminated from dI/dt.

D Versions of Ortocartan for different computers.

As mentioned in the introduction, Ortocartan was originally written and implemented in the University of Texas Lisp 4.1 on a CDC Cyber 73 computer. Those computers were scrapped in all sites where Ortocartan was used on them. The file with the U.T. Lisp code of Ortocartan is still preserved and can be obtained from the author (A. K.), but since I do not have access to the U. T. Lisp myself, this version will not be maintained further, and has probably already become defunct (which is a pity because the U.T. Lisp was a dialect of superb elegance).

The versions of Ortocartan written in an older implementation of Cambridge Lisp for IBM 360/370 computers and in the Slisp/360 version for Siemens 4004 computers have been defunct for some years already. Information is missing on the Lisp 1108 version

for UNIVAC computers, but it was not under our care anyway. You may try to obtain information on it from Dr. Gokturk Ucoluk, Fizik Bolumu, ODTU, Ankara, Turkey.

The Cambridge Lisp version for the Atari Mega ST computers still exists and works, although the algebraic computing community seems to have taken divorce from these computers quite a while ago. It will probably become defunct together with A.K.'s Atari computer that may be among the last ones still working. The previous edition of this manual describes how to use that version, it can be obtained from A. Krasiński.

The only version now under maintenance is the Codemist Standard Lisp version that will run wherever CSL can be used. See Appendix A for more details.

E Sample prints.

In this appendix, we present copies of original outputs from the computer. The examples exhibit the various features of the program.

The printouts are broken into lines without obeying the standard rules of neat printing, such as: do not divide expressions of the form $f(x)$, do not jump to the next line just before the right parenthesis or just after the left parenthesis, do not separate the base from its exponent, and so on. If the printout is to be inserted in a text for publication, then the user is advised to use the (output for latex) command described in Section 23.

This untidiness of Ortocartan printouts is the price we had to pay for making the printing procedure more powerful. Consider the following example (these are copies of parts of the original input and output for the program "calculate", transferred here by a text-editor, with some irrelevant or empty lines removed):

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

The output for this example will be:

```
(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 = 50 msec)
```

You have never seen a problem involving this kind of expression? Well, honestly, neither have we. But you can be safely assured that, no matter how wildly complicated an expression is, ortocartan and calculate will know how to handle and print it. This generality made it difficult to instruct the program to end each line only at a + or a − in the base level, but someday we may come back to the problem and solve it. By the way, try to print the same expression with Latex – we dare say Ortocartan does it in a more readable way.

Each base-level line of the printout begins with the sign >, to facilitate reading those lines that were broken in unusual places. For each example, the input data was read from a disk file.

For each of the examples, the input data as prepared by the user is shown separately. The output is copied from disk files produced according to the instructions from sec. 7.

E.1 Example I: The Robertson-Walker Metric

$$ds^2 = dt^2 - R^2[dr^2/(1-kr^2) + r^2(d\vartheta^2 + \sin^2\vartheta d\varphi^2)],$$

where $R = R(t)$ is an arbitrary function and k is an arbitrary constant.

Reference

W. Rindler, *Essential relativity.* Van Nostrand Reinhold Company, New York-Cincinnati-Toronto-London-Melbourne 1969, p. 234.

The input data is here:

```
(setq !*lower nil)
 (ortocartan '(
    (Robertson-Walker metric)
    (coordinates t r theta phi)
    (functions R (t) )
```

```
    (constants k)
    (ematrix 1 0 0 0 0 (R / (1 - k * r ^ 2) ^ (1 2)) 0 0 0 0
             (r * R) 0 0 0 0 (r * R * (sin theta))  )
     ))
(setq !*lower t)
```

Notes

The metric is conformally flat - please verify that Ortocartan has recognized it. Here we asked the Lisp system to treat upper- and lower case letters as different symbols (the first line of input). The command in the last line of the input reverses the command from the first line.

The output is this (some irrelevant or empty lines were deleted by the text-editor. The irrelevant lines are not parts of Ortocartan's output, but are responses of the Lisp system that are not interesting for the user.):

```
(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 (theta)
             3

(ematrix completed)
(TIME = 190 msec)

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


(DETERMINANT EMATRIX calculated)
(TIME = 260 msec)
```

```
      0
>   ie.   = 1
        0

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

      2      -1  -1
>   ie.   = r   R
        2

      3      -1  -1    -1
>   ie.   = r   R   sin  (theta)
        3

(ie calculated)
(TIME = 400 msec)

          1              -1
>   agamma      = (1/2) R   R,
            0 1               t

          2              -1
>   agamma      = (1/2) R   R,
            0 2               t

          2              -1  -1         2 (1/2)
>   agamma      = (1/2) r   R   (1 - k r )
            1 2

          3              -1
>   agamma      = (1/2) R   R,
            0 3               t

          3              -1  -1         2 (1/2)
>   agamma      = (1/2) r   R   (1 - k r )
            1 3

          3              -1  -1                 -1
>   agamma      = (1/2) r   R   cos (theta) sin  (theta)
            2 3

(agamma calculated)
```

```
(TIME = 590 msec)

(agamma completed)
(TIME = 590 msec)

         0        -1
>   gamma      = R   R,
           1 1         t


         0        -1
>   gamma      = R   R,
           2 2         t


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


         1          -1  -1         2 (1/2)
>   gamma      = - r   R   (1 - k r )
           2 2


         1          -1  -1         2 (1/2)
>   gamma      = - r   R   (1 - k r )
           3 3


         2          -1  -1                -1
>   gamma      = - r   R   cos (theta) sin  (theta)
           3 3


(gamma calculated)
(TIME = 680 msec)

(gamma completed)
(TIME = 690 msec)

                     -1
>   riemann        = R   R,
           0 1 0 1         t t
```

```
                      -1
>   riemann        = R   R,
           0 2 0 2         t t

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

                           -2    -2     2
>   riemann        = - k R   - R   R,
           1 2 1 2                   t

                           -2    -2     2
>   riemann        = - k R   - R   R,
           1 3 1 3                   t

                           -2    -2     2
>   riemann        = - k R   - R   R,
           2 3 2 3                   t

(riemann calculated)
(TIME = 930 msec)

(riemann completed)
(TIME = 980 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 = 1000 msec)
```

```
(CURVATURE INVARIANT calculated)
(TIME = 1000 msec)

                                  -2      -2     2      -1
>   CURVATURE INVARIANT = - 6 k R   - 6 R   R,    - 6 R   R,
                                                t             t t
(weyl calculated)
(TIME = 1090 msec)


(ALL COMPONENTS OF THE WEYL TENSOR ARE ZERO)


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

END OF WORK
(RUN TIME = 1090 msec)
```

E.2 Example II: The most general spherically symmetric metric in the Schwarzschild coordinate system.

$$ds^2 = \mathrm{e}^{2\nu}dt^2 - \mathrm{e}^{2\mu}dr^2 - r^2(d\vartheta^2 + \sin^2\vartheta d\varphi^2),$$

where $\nu = \nu(t, r)$ and $\mu = \mu(t, r)$ are arbitrary functions.

Reference

Most textbooks on general relativity, e.g. J.L. Synge, *Relativity, the general theory.* North Holland Publishing Company, Amsterdam 1960, p. 265.

The input data is here:

```
(ortocartan '(
    (spherically symmetric standard)
    (coordinates t r theta phi)
    (functions nu (t r) mu (t r))
    (ematrix (exp nu) 0 0 0 0 (exp mu) 0 0 0 0
             r 0 0 0 0 (r * (sin theta)))
    (rmargin 80)
            ))
```

Notes

nu stands for ν, mu stands for μ. (rmargin 80) will suit the output to the pagewidth of this text. Note the untidy linebreaks in the Riemann and Ricci tensors. Some irrelevant and empty lines have been deleted again. The output is

```
(spherically symmetric standard)

          0
>   ematrix  . = exp (nu)
            0

          1
>   ematrix  . = exp (mu)
            1

          2
>   ematrix  . = r
            2

          3
>   ematrix  . = r sin (theta)
            3

(ematrix completed)
(TIME = 20 msec)

                           2
>   DETERMINANT EMATRIX = r  exp (nu + mu) sin (theta)


(DETERMINANT EMATRIX calculated)
(TIME = 40 msec)


     0
>   ie.   = exp (- nu)
        0

     1
>   ie.   = exp (- mu)
        1

     2      -1
>   ie.   = r
        2

     3      -1    -1
>   ie.   = r   sin  (theta)
        3
```

```
(ie calculated)
(TIME = 200 msec)


          0
>   agamma      = - (1/2) exp (- mu) nu,
            0 1                        r

          1
>   agamma      = (1/2) exp (- nu) mu,
            0 1                      t

          2              -1
>   agamma      = (1/2) r   exp (- mu)
            1 2

          3              -1
>   agamma      = (1/2) r   exp (- mu)
            1 3

          3              -1                -1
>   agamma      = (1/2) r   cos (theta) sin  (theta)
            2 3

(agamma calculated)
(TIME = 360 msec)

(agamma completed)
(TIME = 360 msec)

         0
>   gamma      = exp (- mu) nu,
           1 0                r


         0
>   gamma      = exp (- nu) mu,
           1 1                t


         1          -1
>   gamma      = - r   exp (- mu)
           2 2
```

```
         1          -1
>   gamma      = - r   exp (- mu)
           3 3


         2          -1                -1
>   gamma      = - r   cos (theta) sin  (theta)
           3 3


(gamma calculated)
(TIME = 380 msec)


(gamma completed)
(TIME = 380 msec)

                                      2
>   riemann        = exp (- 2 nu) mu,    + exp (- 2 nu) mu,     - exp (- 2 nu)
           0 1 0 1                  t                      t t

                                       2
>       nu,   mu,   - exp (- 2 mu) nu,    - exp (- 2 mu) nu,     + exp (- 2 mu)
          t     t                    r                      r r


>       nu,   mu,
          r     r

                        -1
>   riemann        = - r   exp (- 2 mu) nu,
           0 2 0 2                          r

                        -1
>   riemann        = - r   exp (- nu - mu) mu,
           0 2 1 2                            t

                        -1
>   riemann        = - r   exp (- 2 mu) nu,
           0 3 0 3                          r

                        -1
>   riemann        = - r   exp (- nu - mu) mu,
           0 3 1 3                            t
```

```
                        -1
>   riemann        = - r   exp (- 2 mu) mu,
          1 2 1 2                          r

                        -1
>   riemann        = - r   exp (- 2 mu) mu,
          1 3 1 3                          r

                      -2                 -2
>   riemann        = r   exp (- 2 mu) - r
          2 3 2 3

(riemann calculated)
(TIME = 690 msec)


(riemann completed)
(TIME = 740 msec)

                  -1                                          2
>   ricci    = 2 r   exp (- 2 mu) nu,   - exp (- 2 nu) mu,    - exp (- 2 nu) mu
         0 0                          r                    t

                                                             2
>       ,     + exp (- 2 nu) nu,   mu,   + exp (- 2 mu) nu,    + exp (- 2 mu)
         t t                    t     t                    r


>       nu,     - exp (- 2 mu) nu,   mu,
           r r                    r     r

                  -1
>   ricci    = 2 r   exp (- nu - mu) mu,
         0 1                           t

                  -1                                          2
>   ricci    = 2 r   exp (- 2 mu) mu,   + exp (- 2 nu) mu,    + exp (- 2 nu) mu
         1 1                          r                    t

                                                             2
>       ,     - exp (- 2 nu) nu,   mu,   - exp (- 2 mu) nu,    - exp (- 2 mu)
         t t                    t     t                    r
```

```
>       nu,     + exp (- 2 mu) nu,   mu,
          r r                   r     r


                 -2                -1                        -1
>   ricci    = - r   exp (- 2 mu) - r   exp (- 2 mu) nu,   + r   exp (- 2 mu)
         2 2                                          r

                -2
>       mu,   + r
          r

                 -2                -1                        -1
>   ricci    = - r   exp (- 2 mu) - r   exp (- 2 mu) nu,   + r   exp (- 2 mu)
         3 3                                          r

                -2
>       mu,   + r
          r
```

(ricci calculated)
(TIME = 820 msec)

(CURVATURE INVARIANT calculated)
(TIME = 850 msec)

```
                             -2                  -1                          -1
>   CURVATURE INVARIANT = 2 r   exp (- 2 mu) + 4 r   exp (- 2 mu) nu,   - 4 r
                                                                        r

                                                 2
>       exp (- 2 mu) mu,   - 2 exp (- 2 nu) mu,    - 2 exp (- 2 nu) mu,     +
                         r                       t                       t t

                                                   2
>       2 exp (- 2 nu) nu,   mu,   + 2 exp (- 2 mu) nu,    + 2 exp (- 2 mu) nu,
                           t     t                     r

                                                -2
>            - 2 exp (- 2 mu) nu,   mu,   - 2 r
         r r                       r     r
```

```
                          -2                          -1
>   weyl        = - (1/3) r   exp (- 2 mu) + (1/3) r   exp (- 2 mu) nu,   - (1/
       0 1 0 1                                                            r


          -1                                                 2
>      3) r   exp (- 2 mu) mu,   + (1/3) exp (- 2 nu) mu,    + (1/3) exp (- 2
                               r                            t


>      nu) mu,     - (1/3) exp (- 2 nu) nu,   mu,   - (1/3) exp (- 2 mu) nu,
              t t                          t     t                          r

       2
>        - (1/3) exp (- 2 mu) nu,     + (1/3) exp (- 2 mu) nu,   mu,   + (1/3)
                                  r r                          r     r

        -2
>      r


                        -2                          -1
>   weyl        = (1/6) r   exp (- 2 mu) - (1/6) r   exp (- 2 mu) nu,   + (1/6)
       0 2 0 2                                                          r

        -1                                                 2
>      r   exp (- 2 mu) mu,   - (1/6) exp (- 2 nu) mu,    - (1/6) exp (- 2 nu)
                             r                            t

                                                                          2
>      mu,     + (1/6) exp (- 2 nu) nu,   mu,   + (1/6) exp (- 2 mu) nu,    +
          t t                          t     t                          r

                                                                         -2
>      (1/6) exp (- 2 mu) nu,     - (1/6) exp (- 2 mu) nu,   mu,   - (1/6) r
                              r r                          r     r

                        -2                          -1
>   weyl        = (1/6) r   exp (- 2 mu) - (1/6) r   exp (- 2 mu) nu,   + (1/6)
       0 3 0 3                                                          r

        -1                                                 2
>      r   exp (- 2 mu) mu,   - (1/6) exp (- 2 nu) mu,    - (1/6) exp (- 2 nu)
                             r                            t
```

```
                                                                    2
>       mu,     + (1/6) exp (- 2 nu) nu,   mu,   + (1/6) exp (- 2 mu) nu,    +
           t t                           t     t                           r

                                                                          -2
>       (1/6) exp (- 2 mu) nu,     - (1/6) exp (- 2 mu) nu,   mu,   - (1/6) r
                              r r                           r     r

                           -2                          -1
>   weyl        = - (1/6) r   exp (- 2 mu) + (1/6) r   exp (- 2 mu) nu,   - (1/
       1 2 1 2                                                            r

           -1                                                 2
>       6) r   exp (- 2 mu) mu,   + (1/6) exp (- 2 nu) mu,    + (1/6) exp (- 2
                              r                           t


>       nu) mu,     - (1/6) exp (- 2 nu) nu,   mu,   - (1/6) exp (- 2 mu) nu,
              t t                           t     t                           r

        2
>         - (1/6) exp (- 2 mu) nu,     + (1/6) exp (- 2 mu) nu,   mu,   + (1/6)
                                 r r                           r     r

         -2
>       r


                           -2                          -1
>   weyl        = - (1/6) r   exp (- 2 mu) + (1/6) r   exp (- 2 mu) nu,   - (1/
       1 3 1 3                                                            r

           -1                                                 2
>       6) r   exp (- 2 mu) mu,   + (1/6) exp (- 2 nu) mu,    + (1/6) exp (- 2
                              r                           t


>       nu) mu,     - (1/6) exp (- 2 nu) nu,   mu,   - (1/6) exp (- 2 mu) nu,
              t t                           t     t                           r

        2
>         - (1/6) exp (- 2 mu) nu,     + (1/6) exp (- 2 mu) nu,   mu,   + (1/6)
                                 r r                           r     r
```

```
         -2
>       r
```

```
                        -2                          -1
>   weyl        = (1/3) r   exp (- 2 mu) - (1/3) r   exp (- 2 mu) nu,   + (1/3)
        2 3 2 3                                                          r

         -1                                            2
>       r   exp (- 2 mu) mu,   - (1/3) exp (- 2 nu) mu,    - (1/3) exp (- 2 nu)
                              r                          t

                                                                          2
>       mu,     + (1/3) exp (- 2 nu) nu,   mu,   + (1/3) exp (- 2 mu) nu,    +
           t t                           t     t                           r

                                                                         -2
>       (1/3) exp (- 2 mu) nu,     - (1/3) exp (- 2 mu) nu,   mu,   - (1/3) r
                                r r                          r     r
```

```
(weyl calculated)
(TIME = 1430 msec)
```

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

END OF WORK
(RUN TIME = 1430 msec)
```

This example provides a good opportunity to demonstrate the output in the Latex format. Let us run the same example with the data modified as follows:

```
(ortocartan '(
    (spherically symmetric standard)
    (coordinates t r theta phi)
    (functions nu (t r) mu (t r))
    (ematrix (exp nu) 0 0 0 0 (exp mu) 0 0 0 0
             r 0 0 0 0 (r * (sin theta)))
    (output for latex)
    (dont print ematrix determinant ie agamma gamma riemann)
    (stop after ricci)
            ))
```

The command (rmargin ...) is not reasonable here (in fact, with the "(output for latex)" command, Ortocartan would ignore any (rmargin ...) command) because the final layout will be made by Latex anyway. In order to save space we have asked Ortocartan not to print anything before the Ricci tensor, and to stop after calculating the Ricci tensor, i.e. to give up on the scalar curvature and the Weyl tensor. The output was produced in the Latex code, that is first shown verbatim:

```
(spherically symmetric standard)

(ematrix completed)
(TIME = 240 msec)


(DETERMINANT EMATRIX calculated)
(TIME = 270 msec)

(ie calculated)
(TIME = 460 msec)

(agamma calculated)
(TIME = 780 msec)


(agamma completed)
(TIME = 780 msec)

(gamma calculated)
(TIME = 830 msec)

(gamma completed)
(TIME = 830 msec)

(riemann calculated)
(TIME = 1070 msec)


(riemann completed)
(TIME = 1110 msec)

\begin{equation}
ricci_{00} = 2r^{-1}\exp (- 2{\mu}){\nu},_{r} - \exp (- 2{\nu}){{\mu},_{t}}^{2}
$$

$$
- \exp (- 2{\nu}){\mu},_{t t} + \exp (- 2{\nu}){\nu},_{t}{\mu},_{t} + \exp (- 2{\mu}){{\
```

```
$$


$$
{\nu},_{r r} - \exp (- 2{\mu}){\nu},_{r}{\mu},_{r}
\end{equation}

\begin{equation}
ricci_{01} = 2r^{-1}\exp (- {\nu} - {\mu}){\mu},_{t}
\end{equation}

\begin{equation}
ricci_{11} = 2r^{-1}\exp (- 2{\mu}){\mu},_{r} + \exp (- 2{\nu}){{\mu},_{t}}^{2}
$$

$$
+ \exp (- 2{\nu}){\mu},_{t t} - \exp (- 2{\nu}){\nu},_{t}{\mu},_{t} - \exp (- 2{\mu}){{\
$$

$$
{\nu},_{r r} + \exp (- 2{\mu}){\nu},_{r}{\mu},_{r}
\end{equation}

\begin{equation}
ricci_{22} = - r^{-2}\exp (- 2{\mu}) - r^{-1}\exp (- 2{\mu}){\nu},_{r}
$$

$$
+ r^{-1}\exp (- 2{\mu}){\mu},_{r} + r^{-2}
\end{equation}

\begin{equation}
ricci_{33} = - r^{-2}\exp (- 2{\mu}) - r^{-1}\exp (- 2{\mu}){\nu},_{r}
$$

$$
+ r^{-1}\exp (- 2{\mu}){\mu},_{r} + r^{-2}
\end{equation}

(ricci calculated)
(TIME = 1250 msec)


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

```
END OF WORK
(RUN TIME = 1250 msec)
```

Yes, some lines are too long to fit into this page. This output was not meant to be shown to humans, but to be read by Latex. But verbatim is verbatim – we do not cheat. Now the same output will be inserted into this text as part of the Latex code (with the time-messages deleted). Note how Ortocartan has recognized the Greek letters and printed them in Latex's favourite way, and what Latex will now do with them.

$$ricci_{00} = 2r^{-1}\exp(-2\mu)\nu_{,r} - \exp(-2\nu)\mu_{,t}{}^2$$

$$-\exp(-2\nu)\mu_{,tt} + \exp(-2\nu)\nu_{,t}\,\mu_{,t} + \exp(-2\mu)\nu_{,r}{}^2 + \exp(-2\mu)$$

$$\nu_{,rr} - \exp(-2\mu)\nu_{,r}\,\mu_{,r} \tag{18}$$

$$ricci_{01} = 2r^{-1}\exp(-\nu-\mu)\mu_{,t} \tag{19}$$

$$ricci_{11} = 2r^{-1}\exp(-2\mu)\mu_{,r} + \exp(-2\nu)\mu_{,t}{}^2$$

$$+\exp(-2\nu)\mu_{,tt} - \exp(-2\nu)\nu_{,t}\,\mu_{,t} - \exp(-2\mu)\nu_{,r}{}^2 - \exp(-2\mu)$$

$$\nu_{,rr} + \exp(-2\mu)\nu_{,r}\,\mu_{,r} \tag{20}$$

$$ricci_{22} = -r^{-2}\exp(-2\mu) - r^{-1}\exp(-2\mu)\nu_{,r}$$

$$+r^{-1}\exp(-2\mu)\mu_{,r} + r^{-2} \tag{21}$$

$$ricci_{33} = -r^{-2}\exp(-2\mu) - r^{-1}\exp(-2\mu)\nu_{,r}$$

$$+r^{-1}\exp(-2\mu)\mu_{,r} + r^{-2} \tag{22}$$

(The equation numbers above are continued from Section 1 because we have not readjusted the equation counter. You can do it in your own Latex preamble.)

E.3 Example III: The Stephani solution.

$$ds^2 = D^2dt^2 - (R/V)^2(dx^2 + dy^2 + dz^2),$$

where

$$V = 1 + \frac{1}{4}k[(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2],$$

$$D = F(V_{,t}/V - R_{,t}/R),$$

R_0, k, x_0, y_0, z_0 and F are arbitrary functions of t.

Reference:

H. Stephani, *Commun. Math. Phys.* **4**, 137 (1967).

The input data is here:

```
(setq !*lower nil)
(ortocartan '(
    (the Stephani solution)
    (coordinates t x y z)
    (functions C (t) k (t) R (t) X0(t) Y0(t) Z0(t) F (t)
               V (t x y z)  D (t x y z)  )
    (ematrix D  0  0  0  0  (R / V)  0  0  0  0  (R / V)
                0  0  0  0  (R / V)  )
    (tensors einstein)
    (markers m)
    (substitutions
    riemann (0 1 0 1) (0 2 0 2) (0 3 0 3)
 ((der t D)*(der t V)) = ((der t D) * V * (D / F + (der t R) / R))
    agamma riemann
 (der m D) = (deriv m (F * ((der t V) / V - (der t R)/ R)))
    riemann
 (der t R) = (R * ((der t V) / V - D / F))
    riemann
 ((der z V) ^ 2) = ((V - 1) * k - (der x V) ^ 2 - (der y V) ^ 2)
    riemann
 ((der t V) / V) = (D / F + (der t R) / R)
    riemann
 V = (1 + (1 4) * k * ((x - X0) ^ 2 + (y - Y0) ^ 2 + (z - Z0) ^ 2))
    riemann
 k = ((C ^ 2 - 1 / F ^ 2) * R ^ 2)
                     )
    (rmargin 61)
    (dont print messages agamma)
                   ))
  (setq !*lower t)
```

Notes:

This is the most general conformally flat nonstatic perfect fluid solution of Einstein's equations (see Kramer et al., *Exact solutions of Einstein's field equations*, Cambridge University Press 1980, p. 371, theorem 32.15). The matter density in it, denoted $3C^2(t)$ here, is a function of t only and is related to k, F and $R$by:

$$k = (C^2 - 1/F^2)R^2.$$

The Einstein tensor is calculated here in addition to the other quantities, and the messages about unsuccessful attempts at substitutions are suppressed. Also, printing the antisymmetrized Ricci rotation coefficients agamma is suppressed. In this example, instead of substituting the explicit forms of the components of the metric tensor, it is more reasonable to feed the information about them into the formulae step by step. Each substitution results in a partial simplification of the formulae. The fourth substitution stems from the identity:

$$V_{,x}{}^2 + V_{,y}{}^2 + V_{,z}{}^2 = k(V - 1).$$

The second substitution makes use of the marker m, it results in replacing $D_{,x}$, $D_{,y}$ and $D_{,z}$ by the appropriate derivatives of $[F(V_{,t}/V - R_{,t}/R)]$.

The first, third and fifth substitutions use the definition of D, but apply it only in certain contexts. For example, in the first substitution, $V_{,t}$ is replaced by $V(D/F+R_{,t}/R)$, but only in those instances where $V_{,t}$ is multiplied by $D_{,t}$.

Each consecutive substitution was guessed after inspecting the output obtained without it. You are encouraged to repeat this procedure. Run this example first without any substitutions, then add the first one and see what has changed, then add the second one, and so on. If you are clever with using your editor, then you may use it to cut the data for this example out of the file containing this manual.

The output is here (again, with irrelevant lines deleted):

```
(the Stephani solution)

substitutions

riemann (0 1 0 1) (0 2 0 2) (0 3 0 3)

                  -1                -1
>   V,   D,   = R   V R,   D,   + F   V D D,
      t    t            t    t             t

agamma riemann

                -1         -1
>   D,   = (F V   V,   - R   F R,  ),
      m             t            t   m
```

```
riemann

                -1        -1
>   R,   = - R F   D + R V   V,
      t                       t

riemann

        2                   2       2
>   V,    = - k + k V - V,    - V,
      z                   x       y

riemann

     -1         -1         -1
>   V   V,   = R   R,   + F   D
          t          t

riemann


>   V = 1 - (1/2) x k X0 - (1/2) y k Y0 - (1/2) z k Z0 + (1/


               2             2             2          2
>       4) k X0  + (1/4) k Y0  + (1/4) k Z0  + (1/4) x  k +


               2           2
>       (1/4) y  k + (1/4) z  k


riemann

         2  2    2  -2
>   k = C  R  - R  F


           0
>   ematrix  . = D
             0

           1        -1
>   ematrix  . = R V
             1
```

```
           2        -1
>   ematrix  . = R V
             2

           3        -1
>   ematrix  . = R V
             3

(ematrix completed)
(TIME = 500 msec)

                           3  -3
>   DETERMINANT EMATRIX = R  V   D


(DETERMINANT EMATRIX calculated)
(TIME = 520 msec)


      0      -1
>   ie.   = D
        0

      1      -1
>   ie.   = R   V
        1

      2      -1
>   ie.   = R   V
        2

      3      -1
>   ie.   = R   V
        3

(ie calculated)
(TIME = 760 msec)

(agamma calculated)
(TIME = 1060 msec)

(agamma completed)
(TIME = 1060 msec)
```

```
         0          -1    -1  -1              -1    -1
>   gamma      = - R   F V   D   V,   V,   + R   F D   V,
          1 0                    t    x               t

>
        x
```

Oops! This is very untidy printing – the t and x in the second derivative of V have been separated. But you can easily learn to live with this, and use the (output for latex) when you need a really neat printout.

```
         0        -1  -1         -1  -1
>   gamma      = R   D   R,   - V   D   V,
          1 1             t              t
```

```
         0          -1    -1  -1              -1    -1
>   gamma      = - R   F V   D   V,   V,   + R   F D   V,
          2 0                    t    y               t

>
        y
```

```
         0        -1  -1         -1  -1
>   gamma      = R   D   R,   - V   D   V,
          2 2             t              t
```

```
         0          -1    -1  -1              -1    -1
>   gamma      = - R   F V   D   V,   V,   + R   F D   V,
          3 0                    t    z               t

>
        z
```

```
         0        -1  -1         -1  -1
>   gamma      = R   D   R,   - V   D   V,
          3 3             t              t
```

```
          1          -1
>   gamma      = - R   V,
          2 1           y


          1         -1
>   gamma      = R   V,
          2 2         x


          1          -1
>   gamma      = - R   V,
          3 1           z


          1         -1
>   gamma      = R   V,
          3 3         x


          2          -1
>   gamma      = - R   V,
          3 2           z


          2         -1
>   gamma      = R   V,
          3 3         y


(gamma calculated)
(TIME = 1160 msec)

(gamma completed)
(TIME = 1170 msec)

                             -1          2
>   riemann        = - C F D   C,   + C
          0 1 0 1               t

                             -1          2
>   riemann        = - C F D   C,   + C
          0 2 0 2               t
```

```
                              -1         2
>   riemann        = - C F D   C,   + C
          0 3 0 3              t

                        2
>   riemann        = - C
          1 2 1 2

                        2
>   riemann        = - C
          1 3 1 3

                        2
>   riemann        = - C
          2 3 2 3

(riemann calculated)
(TIME = 5931 msec)

(riemann completed)
(TIME = 6001 msec)

                      -1           2
>   ricci    = 3 C F D   C,   - 3 C
         0 0              t

                      -1           2
>   ricci    = - C F D   C,   + 3 C
         1 1              t

                      -1           2
>   ricci    = - C F D   C,   + 3 C
         2 2              t

                      -1           2
>   ricci    = - C F D   C,   + 3 C
         3 3              t

(ricci calculated)
(TIME = 6021 msec)

(CURVATURE INVARIANT calculated)
(TIME = 6031 msec)
```

```
                                  -1            2
>   CURVATURE INVARIANT = 6 C F D   C,   - 12 C
                                    t


                     2
>   einstein    = 3 C
            0 0


                          -1           2
>   einstein    = 2 C F D   C,   - 3 C
            1 1               t


                          -1           2
>   einstein    = 2 C F D   C,   - 3 C
            2 2               t


                          -1           2
>   einstein    = 2 C F D   C,   - 3 C
            3 3               t


(einstein calculated)
(TIME = 6061 msec)

(weyl calculated)
(TIME = 6141 msec)

(ALL COMPONENTS OF THE WEYL TENSOR ARE ZERO)

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

END OF WORK
(RUN TIME = 6141 msec)
```

E.4 Example IV: The Nariai solution.

$$ds^2 = P^2 dt^2 - (L^2/r^2)(dx^2 + dy^2 + dz^2),$$

where $P = a(t)\cos(\log(r/L)) + b(t)\sin(\log(r/L))$, L is a constant, $r^2 = x^2 + y^2 + z^2$, $a(t)$ and $b(t)$ are arbitrary functions of time.

Reference:

A. Krasiński, J. Plebański, *Rep. Math. Phys.* **17**, 217 (1980).
The input data is:

```
(setq !*lower nil)
 (ortocartan '(
    (Nariai solution)
    (coordinates t x y z)
    (constants L)
    (markers m)
    (ematrix P   0   0   0
        0   (L / r)   0   0
        0   0   (L / r)   0
        0   0   0   (L / r))
    (symbols
      P = (a * (cos (log (r / L))) + b * (sin (log (r / L)))) )
    (substitutions
    agamma riemann
     (der m r) = (m / r)
    agamma riemann
     (cos (log (r / L))) = (P / a - b * (sin (log (r / L)))/ a)
    agamma riemann
     (z ^ 2) = (r ^ 2 - x ^ 2 - y ^ 2)
            )
    (functions a (t) b (t) r(x y z) )
    (dont print messages)
    (rmargin 61)
          ))
(setq !*lower t)
```

Notes

This is a solution of the Einstein's equations in empty space with the cosmological term (note the form of the Ricci tensor in the output!). Note the use of markers which implies substitutions by pattern-matching, they are simpler than in the previous example, so may be more readable. It was not specified here in which components of agamma and riemann the substitutions should be performed. The command (dont print messages) suppresses the information about unsuccessful attempts at substitutions.

The output is:

```
(Nariai solution)

symbols

>   P = a cos (- log (L) + log (r)) + b sin (- log (L) + log
```

```
>       (r))

substitutions

agamma riemann

              -1
>   r,   = m r
      m


agamma riemann

                                      -1
>   cos (- log (L) + log (r)) = - a   b sin (- log (L) + log


                -1
>       (r)) + a   P


agamma riemann

     2      2    2    2
>   z  = - x  - y  + r


           0
>   ematrix  . = P
             0

           1        -1
>   ematrix  . = L r
             1

           2        -1
>   ematrix  . = L r
             2

           3        -1
>   ematrix  . = L r
             3

(ematrix completed)
```

```
(TIME = 300 msec)


                           3  -3
>   DETERMINANT EMATRIX = L  r   P


(DETERMINANT EMATRIX calculated)
(TIME = 340 msec)


      0      -1
>   ie.   = P
        0

      1      -1
>   ie.   = L   r
        1

      2      -1
>   ie.   = L   r
        2

      3      -1
>   ie.   = L   r
        3

(ie calculated)
(TIME = 490 msec)


          0              -1    -1    -1
>   agamma      = (1/2) L   x r   a P   sin (- log (L) + log
            0 1

                       -1    -1  -1            -1    -1  -1
>       (r)) - (1/2) L   x r   a   b + (1/2) L   x r   a   b


        2  -1
>         P   sin (- log (L) + log (r))
```

```
          0              -1    -1    -1
>   agamma      = (1/2) L   y r   a P   sin (- log (L) + log
            0 2


                       -1    -1  -1            -1    -1  -1
>       (r)) - (1/2) L   y r   a   b + (1/2) L   y r   a   b


        2  -1
>         P   sin (- log (L) + log (r))


          0              -1    -1    -1
>   agamma      = (1/2) L   z r   a P   sin (- log (L) + log
            0 3

                       -1    -1  -1            -1    -1  -1
>       (r)) - (1/2) L   z r   a   b + (1/2) L   z r   a   b


        2  -1
>         P   sin (- log (L) + log (r))


          1              -1    -1
>   agamma      = (1/2) L   y r
            1 2

          1              -1    -1
>   agamma      = (1/2) L   z r
            1 3

          2                -1    -1
>   agamma      = - (1/2) L   x r
            1 2

          2              -1    -1
>   agamma      = (1/2) L   z r
            2 3

          3                -1    -1
>   agamma      = - (1/2) L   x r
            1 3
```

```
          3                -1    -1
>   agamma      = - (1/2) L   y r
           2 3

(agamma calculated)
(TIME = 850 msec)

(agamma completed)
(TIME = 860 msec)

         0          -1    -1    -1
>   gamma      = - L   x r   a P   sin (- log (L) + log (r))
          1 0

           -1    -1  -1      -1    -1  -1  2  -1
>       + L   x r   a   b - L   x r   a   b  P   sin (- log


>       (L) + log (r))



         0          -1    -1    -1
>   gamma      = - L   y r   a P   sin (- log (L) + log (r))
          2 0

           -1    -1  -1      -1    -1  -1  2  -1
>       + L   y r   a   b - L   y r   a   b  P   sin (- log


>       (L) + log (r))



         0          -1    -1    -1
>   gamma      = - L   z r   a P   sin (- log (L) + log (r))
          3 0

           -1    -1  -1      -1    -1  -1  2  -1
>       + L   z r   a   b - L   z r   a   b  P   sin (- log
```

```
>       (L) + log (r))



         1          -1    -1
>   gamma      = - L   y r
           2 1


         1         -1    -1
>   gamma      = L   x r
           2 2


         1          -1    -1
>   gamma      = - L   z r
           3 1


         1         -1    -1
>   gamma      = L   x r
           3 3


         2          -1    -1
>   gamma      = - L   z r
           3 2


         2         -1    -1
>   gamma      = L   y r
           3 3


(gamma calculated)
(TIME = 970 msec)

(gamma completed)
(TIME = 980 msec)

                      -2  2  -2
>   riemann        = L   x  r
           0 1 0 1
```

```
                    -2      -2
>   riemann        = L   x y r
          0 1 0 2


                    -2      -2
>   riemann        = L   x z r
          0 1 0 3

                    -2  2  -2
>   riemann        = L   y  r
          0 2 0 2

                    -2      -2
>   riemann        = L   y z r
          0 2 0 3

                      -2  2  -2    -2  2  -2    -2
>   riemann        = - L   x  r   - L   y  r   + L
          0 3 0 3

                    -2  2  -2    -2  2  -2    -2
>   riemann        = L   x  r   + L   y  r   - L
          1 2 1 2

                    -2      -2
>   riemann        = L   y z r
          1 2 1 3

                      -2      -2
>   riemann        = - L   x z r
          1 2 2 3

                      -2  2  -2
>   riemann        = - L   y  r
          1 3 1 3

                    -2      -2
>   riemann        = L   x y r
          1 3 2 3

                      -2  2  -2
>   riemann        = - L   x  r
          2 3 2 3
```

```
(riemann calculated)
(TIME = 3260 msec)

(riemann completed)
(TIME = 3300 msec)

                 -2
>   ricci    = - L
         0 0

                -2
>   ricci    = L
         1 1

                -2
>   ricci    = L
         2 2

                -2
>   ricci    = L
         3 3

(ricci calculated)
(TIME = 3330 msec)

(CURVATURE INVARIANT calculated)
(TIME = 3330 msec)

                                -2
>   CURVATURE INVARIANT = - 4 L



                   -2  2  -2          -2
>   weyl         = L   x  r   - (1/3) L
        0 1 0 1

                   -2      -2
>   weyl         = L   x y r
        0 1 0 2

                   -2      -2
>   weyl         = L   x z r
        0 1 0 3
```

```
                  -2  2  -2          -2
>   weyl        = L   y  r   - (1/3) L
        0 2 0 2

                  -2      -2
>   weyl        = L   y z r
        0 2 0 3

                    -2  2  -2    -2  2  -2          -2
>   weyl        = - L   x  r   - L   y  r   + (2/3) L
        0 3 0 3

                  -2  2  -2    -2  2  -2          -2
>   weyl        = L   x  r   + L   y  r   - (2/3) L
        1 2 1 2

                  -2      -2
>   weyl        = L   y z r
        1 2 1 3

                    -2      -2
>   weyl        = - L   x z r
        1 2 2 3

                    -2  2  -2          -2
>   weyl        = - L   y  r   + (1/3) L
        1 3 1 3

                  -2      -2
>   weyl        = L   x y r
        1 3 2 3

                    -2  2  -2          -2
>   weyl        = - L   x  r   + (1/3) L
        2 3 2 3

(weyl calculated)
(TIME = 3420 msec)


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

END OF WORK
(RUN TIME = 3420 msec)
```

E.5 Example V: The Laplace equation in the cylindrical coordinates.

The input data is here:

```
(calculate '(
   (Laplace equation in cylindrical coordinates)
   (coordinates x y z)
   (functions F (r phi z))
   (symbols
r = ((x ^ 2 + y ^ 2) ^ (1 2))
phi = (arctan (y / x))
cosphi = (cos phi)
      )
   (substitutions
(x ^ 2) = (r ^ 2 - y ^ 2)
(1 + y ^ 2 / x ^ 2) = (cosphi ^ -2)
x = (r * cosphi)
y = (r * (sin phi))
((sin phi) ^ 3) = ((sin phi) * (1 - cosphi ^ 2))
cosphi = (cos phi)
   )
(operation ((deriv x x F) + (deriv y y F) + (deriv z z F)))
   ))
```

Notes

This example shows how to use the program "calculate". Since coordinate transformations as such are not available in Ortocartan, the transformation to the cylindrical coordinates is achieved through a trick: the unknown function F is defined to depend not directly on the cartesian coordinates x and y, but on r and ϕ which are defined as explicit expressions in x and y. In the result thus obtained, x and y are replaced by the appropriate functions of r and ϕ. Note how the program was prevented from replacing $\cos^2\phi$ by $(1-\sin^2\phi)$ at a too early stage of the calculation (this last trick is described in sec. 18).

The output is:

```
(laplace equation in cylindrical coordinates)

symbols

          2    2 (1/2)
>   r = (x  + y )
```

```
                  -1
>   phi = arctan (x   y)

>   cosphi = cos (phi)

substitutions

everywhere

     2      2    2
>   x  = - y  + r

everywhere

         -2  2         -2
>   1 + x   y  = cosphi


everywhere

>   x = r cosphi


everywhere

>   y = r sin (phi)


everywhere

       3                2
>   sin (phi) = - cosphi  sin (phi) + sin (phi)


everywhere

>   cosphi = cos (phi)


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

>       deriv (x,x,f) + deriv (y,y,f) + deriv (z,z,f)


THE RESULT IS
```

```
                -2                -1
>   result  = r   f,          + r   f,    + f,      + f,
          1          phi phi           r       z z       r r


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

END OF WORK (RUN TIME = 550 msec)
```

E.6 Example VI: The spherically symmetric metric in the standard coordinates with the arguments of functions written out explicitly.

The input data here is:

```
(setq !*lower nil)
 (ortocartan '(
     (SPHERICAL WITH ARGUMENTS)
     (coordinates T R THETA PHI)
     (functions MU (T R) NU (T R))
     (ematrix (exp (NU T R))  0  0  0  0  (exp (MU T R))  0
              0  0  0  R  0  0  0  0  (R * (sin THETA))  )
     (dont print ie agamma riemann)
     (stop after ricci)
     (rmargin 61)
          ))
(setq !*lower t)
```

Notes

This example is in fact a duplicate copy of the example II, it is only meant to show that, if the user wishes so, then the arguments of functional expressions can be written out explicitly. Most of the output is suppressed.

The output is:

```
(SPHERICAL WITH ARGUMENTS)


           0
>   ematrix  . = exp (NU (T,R))
             0
```

```
          1
>   ematrix  . = exp (MU (T,R))
            1

          2
>   ematrix  . = R
            2

          3
>   ematrix  . = R sin (THETA)
            3

(ematrix completed)
(TIME = 50 msec)

                           2
>   DETERMINANT EMATRIX = R  exp (MU (T,R) + NU (T,R)) sin (


>       THETA)


(DETERMINANT EMATRIX calculated)
(TIME = 90 msec)

(ie calculated)
(TIME = 180 msec)

(agamma calculated)
(TIME = 290 msec)

(agamma completed)
(TIME = 300 msec)

         0
>   gamma      = exp (- MU (T,R)) (NU,   (T,R))
           1 0                          2


         0
>   gamma      = exp (- NU (T,R)) (MU,   (T,R))
           1 1                          1
```

```
         1          -1
>   gamma      = - R   exp (- MU (T,R))
           2 2
```

```
         1          -1
>   gamma      = - R   exp (- MU (T,R))
           3 3
```

```
         2          -1                -1
>   gamma      = - R   cos (THETA) sin  (THETA)
           3 3
```

```
(gamma calculated)
(TIME = 340 msec)

(gamma completed)
(TIME = 340 msec)

(riemann calculated)
(TIME = 530 msec)

(riemann completed)
(TIME = 560 msec)
```

```
                 -1
>   ricci    = 2 R   exp (- 2 MU (T,R)) (NU,   (T,R)) + exp
         0 0                                 2

                                    2
>       (- 2 MU (T,R)) (NU,   (T,R))  + exp (- 2 MU (T,R)) (
                           2
```

```
>       NU,     (T,R)) - exp (- 2 MU (T,R)) (MU,   (T,R)) (
           2 2                                   2

                                                      2
>       NU,   (T,R)) - exp (- 2 NU (T,R)) (MU,   (T,R))  -
           2                                     1
```

```
>       exp (- 2 NU (T,R)) (MU,     (T,R)) + exp (- 2 NU (T,
                                1 1

>       R)) (MU,   (T,R)) (NU,   (T,R))
                1             1

                  -1
>   ricci    = 2 R   exp (- MU (T,R) - NU (T,R)) (MU,   (T,R
        0 1                                           1

>       ))

                  -1
>   ricci    = 2 R   exp (- 2 MU (T,R)) (MU,   (T,R)) - exp
        1 1                                   2

                                    2
>       (- 2 MU (T,R)) (NU,   (T,R))  - exp (- 2 MU (T,R)) (
                           2

>       NU,     (T,R)) + exp (- 2 MU (T,R)) (MU,   (T,R)) (
           2 2                                      2

                                                        2
>       NU,   (T,R)) + exp (- 2 NU (T,R)) (MU,   (T,R))  +
           2                                       1

>       exp (- 2 NU (T,R)) (MU,     (T,R)) - exp (- 2 NU (T,
                                1 1

>       R)) (MU,   (T,R)) (NU,   (T,R))
                1             1

                  -2                          -1
>   ricci    = - R   exp (- 2 MU (T,R)) + R   exp (- 2 MU (T
        2 2
```

```
                                -1
>       ,R)) (MU,   (T,R)) - R   exp (- 2 MU (T,R)) (NU,   (
                 2                                      2


                 -2
>       T,R)) + R


                  -2                          -1
>   ricci    = - R   exp (- 2 MU (T,R)) + R   exp (- 2 MU (T
         3 3

                                -1
>       ,R)) (MU,   (T,R)) - R   exp (- 2 MU (T,R)) (NU,   (
                 2                                      2

                 -2
>       T,R)) + R


(ricci calculated)
(TIME = 689 msec)


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

END OF WORK
(RUN TIME = 689 msec)
```

E.7 Example VII: Application of the program Ellisevol to check the Ellis evolution equations for the Lanczos metric.

The Lanczos metric is:

$$ds^2 = (dt + Crd\varphi)^2 - \psi d\varphi^2 - \frac{1}{4}\mathrm{e}^{-r}dr^2/\psi - \mathrm{e}^{-r}dz^2,$$

where

$$\psi = (C^2 r + \Lambda - \Lambda \mathrm{e}^{-r}).$$

References

The original paper: K. Lanczos, *Zeitschrift für Physik* **21**, 73 (1924).
English translation: *Gen. Rel. Grav.* **29**, 363 (1997).
The input data is here:

```
 (setq !*lower nil)
 (ellisevol'(
    (LANCZOS METRIC)
    (coordinates t phi r z)
    (velocity 1 0 0 0)
    (constants C Lambda)
    (symbols psi = (C ^ 2 * r + Lambda - Lambda * (exp (-  r))) )
    (ematrix  1  (C * r)  0  0
              0  ((C ^ 2 * r + Lambda - Lambda * (exp (-  r)))^ (1 2 ))
          0 0 0 0 ((1 2) * (exp ((-1 2) * r)) * (C ^ 2 * r + Lambda
                    - Lambda * (exp (- r))) ^ (-1 2))  0
              0  0  0  (exp (- (1 2) * r)))
 (substitutions (C ^ 2 * r + Lambda - Lambda * (exp (-  r))) = psi )
   (dont print messages)
   (tensors einstein)
         ))
(setq !*lower t)
```

Notes

This is a stationary cylindrically symmetric solution of Einstein's equations with a rotating dust source and with a nonvanishing cosmological constant Λ. The coordinates used in the metric shown above are comoving and the velocity vector field of the dust is one of the orthonormal tetrad vectors, hence the tetrad components of velocity field are `(1 0 0 0)`. Since this is a solution of Einstein's equations, this vector field is uniquely determined by the metric, and so, as expected, all the constraint and evolution equations will be identities. However, the acceleration (= 0), rotation, expansion (= 0), and shear (= 0) are all calculated, along with the electric and magnetic parts of the Weyl tensor. This example is well suited to try out the (output for latex) option – try it yourself. The substitution in line 5 from the bottom was requested to be done everywhere – this is usually not a reasonable option because it makes the calculation slower. It was done this way here in order to use it together with the (dont print messages) option that suppresses all the messages about unsuccessful attempts at substitutions. The Einstein tensor calculated along the way makes it possible to easily see that it is a dust solution indeed.

```
(LANCZOS METRIC)

symbols
                                        2
>   psi = Lambda - Lambda exp (- r) + C  r

substitutions

everywhere
```

```
                                  2
>   Lambda - Lambda exp (- r) + C  r = psi




           0
>   ematrix  . = 1
             0

           0
>   ematrix  . = C r
             1

           1        (1/2)
>   ematrix  . = psi
             1

           2                                 - (1/2)
>   ematrix  . = (1/2) exp (- (1/2) r) psi
             2

           3
>   ematrix  . = exp (- (1/2) r)
             3

(ematrix completed)
(TIME = 210 msec)


            0
>   velocity  = 1


>   DETERMINANT EMATRIX = (1/2) exp (- r)


(DETERMINANT EMATRIX calculated)
(TIME = 270 msec)


      0
>   ie.   = 1
        0
```

```
      0              - (1/2)
>   ie.   = - C r psi
        1

      1        - (1/2)
>   ie.   = psi
        1

      2                         (1/2)
>   ie.   = 2 exp ((1/2) r) psi
        2

      3
>   ie.   = exp ((1/2) r)
        3

(ie calculated)
(TIME = 450 msec)


         0
>   uvelo  = 1



>   lvelo  = 1
         0


>   lvelo  = C r
         1



>   metric    = 1
          0 0


>   metric    = C r
          0 1


                        2  2
>   metric    = - psi + C  r
          1 1
```

```
                                      -1
>   metric    = - (1/4) exp (- r) psi
          2 2


>   metric    = - exp (- r)
          3 3


(metric calculated)
(TIME = 520 msec)


             0 0        2  2    -1
>   invmetric    = 1 - C  r  psi



             0 1          -1
>   invmetric    = C r psi



             1 1        -1
>   invmetric    = - psi



             2 2
>   invmetric    = - 4 exp (r) psi



             3 3
>   invmetric    = - exp (r)


(invmetric calculated)
(TIME = 560 msec)

          0
>   agamma      = - C exp ((1/2) r)
             1 2
```

```
          1                                          - (1/2)          2
>   agamma      = - (1/2) Lambda exp (- (1/2) r) psi        - (1/2) C
            1 2

                          - (1/2)
>       exp ((1/2) r) psi


          3                                  (1/2)
>   agamma      = - (1/2) exp ((1/2) r) psi
            2 3

(agamma calculated)
(TIME = 750 msec)

(agamma completed)
(TIME = 760 msec)

         0
>   gamma      = - C exp ((1/2) r)
           1 2


         0
>   gamma      = C exp ((1/2) r)
           2 1


         1
>   gamma      = - C exp ((1/2) r)
           2 0


         1                                  - (1/2)    2
>   gamma      = Lambda exp (- (1/2) r) psi        + C  exp ((1/2) r)
           2 1

           - (1/2)
>       psi


         2                         (1/2)
>   gamma      = exp ((1/2) r) psi
           3 3
```

```
(gamma calculated)
(TIME = 810 msec)

(gamma completed)
(TIME = 810 msec)

               0              2      -1
>   christoffel      = (1/2) C  r psi
                 0 2


               0                                                  -1
>   christoffel      = (1/2) C - (1/2) C Lambda r exp (- r) psi   - (1/
                 1 2

            3      -1          3  2    -1
>       2) C  r psi   + (1/2) C  r  psi


               1                     -1
>   christoffel      = - (1/2) C psi
                 0 2


               1                                  -1          2      -1
>   christoffel      = (1/2) Lambda exp (- r) psi   - (1/2) C  r psi
                 1 2

                 2    -1
>       + (1/2) C  psi


               2
>   christoffel      = 2 C exp (r) psi
                 0 1


               2                            2                     2
>   christoffel      = - 2 Lambda psi + 4 C  r exp (r) psi - 2 C  exp (
                 1 1


>       r) psi
```

```
                2                                                  -1
>   christoffel      = - (1/2) - (1/2) Lambda exp (- r) psi    - (1/2) C
                  2 2

        2    -1
>         psi


                2
>   christoffel      = 2 psi
                  3 3


                3
>   christoffel      = - (1/2)
                  2 3


(CHRISTOFFEL SYMBOLS calculated)
(TIME = 1000 msec)

(CHRISTOFFEL SYMBOLS completed)
(TIME = 1000 msec)


>   vtida      = (1/2) C
         1 ; 2


>   vtida      = - (1/2) C
         2 ; 1


((TIDAL MATRIX OF lvelo) completed)
(TIME = 1050 msec)


ACCELERATION = 0


>   rotdd    = (1/2) C
         1 2
```

```
(rotdd calculated)
(TIME = 1060 msec)

(rotdd completed)
(TIME = 1070 msec)

           2
>   rotdu    = - 2 C exp (r) psi
         1

           0            2      -1
>   rotdu    = - (1/2) C  r psi
         2

           1              -1
>   rotdu    = (1/2) C psi
         2

(rotdu completed)
(TIME = 1100 msec)


                        2
>   ROTATION SQUARED = C  exp (r)


(ROTATION SCALAR calculated)
(TIME = 1100 msec)


>   projdd    = - psi
          1 1


                                      -1
>   projdd    = - (1/4) exp (- r) psi
          2 2


>   projdd    = - exp (- r)
          3 3


(projdd calculated)
(TIME = 1110 msec)
```

```
(projdd completed)
(TIME = 1110 msec)

            0
>   projdu    = - C r
          1

            1
>   projdu    = 1
          1

            2
>   projdu    = 1
          2

            3
>   projdu    = 1
          3

(projdu calculated)
(TIME = 1140 msec)


>   EXPANSION SCALAR = 0


(EXPANSION SCALAR calculated)
(TIME = 1150 msec)

(sheardd calculated)
(TIME = 1160 msec)

SHEAR = 0

(lacce IS COVARIANTLY CONSTANT)
((TIDAL MATRIX OF lacce) completed)
(TIME = 1160 msec)

(ALL THE ROTATION CONSTRAINTS ARE FULFILLED IDENTICALLY)
(ROTATION CONSTRAINTS calculated)
(TIME = 1160 msec)

(ALL THE SHEAR CONSTRAINTS ARE FULFILLED IDENTICALLY)
(SHEAR CONSTRAINTS calculated)
```

```
(TIME = 2500 msec)

(ALL THE ROTATION EVOLUTION EQUATIONS ARE FULFILLED IDENTICALLY)
(ROTATION EVOLUTION EQUATIONS calculated)
(TIME = 2800 msec)

                        2
>   riemann        = - C  exp (r)
           0 1 0 1

                        2
>   riemann        = - C  exp (r)
           0 2 0 2

                                  (1/2)
>   riemann        = - C exp (r) psi
           0 2 1 2

                                 (1/2)
>   riemann        = C exp (r) psi
           0 3 1 3

                                   2
>   riemann        = - Lambda - 2 C  exp (r)
           1 2 1 2

                                  2
>   riemann        = - Lambda - C  exp (r)
           1 3 1 3

                                  2
>   riemann        = - Lambda - C  exp (r)
           2 3 2 3

(riemann calculated)
(TIME = 3050 msec)

(riemann completed)
(TIME = 3090 msec)

                  2
>   ricci    = 2 C  exp (r)
         0 0
```

```
                                2
>   ricci    = 2 Lambda + 2 C  exp (r)
         1 1


                                2
>   ricci    = 2 Lambda + 2 C  exp (r)
         2 2

                                2
>   ricci    = 2 Lambda + 2 C  exp (r)
         3 3

(ricci calculated)
(TIME = 3120 msec)

(CURVATURE INVARIANT calculated)
(TIME = 3120 msec)

                                            2
>   CURVATURE INVARIANT = - 6 Lambda - 4 C  exp (r)


                                  2
>   einstein    = 3 Lambda + 4 C  exp (r)
            0 0


>   einstein    = - Lambda
            1 1


>   einstein    = - Lambda
            2 2


>   einstein    = - Lambda
            3 3

(einstein calculated)
(TIME = 3140 msec)

>   RAYCHAUDHURI EQUATION = 0

(RAYCHAUDHURI EQUATION calculated)
```

```
(TIME = 3150 msec)

                              2
>   weyl        = - (1/3) C  exp (r)
        0 1 0 1

                              2
>   weyl        = - (1/3) C  exp (r)
        0 2 0 2

                                    (1/2)
>   weyl        = - C exp (r) psi
        0 2 1 2

                             2
>   weyl        = (2/3) C  exp (r)
        0 3 0 3

                                  (1/2)
>   weyl        = C exp (r) psi
        0 3 1 3

                              2
>   weyl        = - (2/3) C  exp (r)
        1 2 1 2

                             2
>   weyl        = (1/3) C  exp (r)
        1 3 1 3

                             2
>   weyl        = (1/3) C  exp (r)
        2 3 2 3

(weyl calculated)
(TIME = 3250 msec)

                             2
>   elweyl    = - (1/3) C  exp (r) psi
          1 1

                              2    -1
>   elweyl    = - (1/12) C  psi
          2 2
```

```
                      2
>   elweyl    = (2/3) C
          3 3


(elweyl calculated)
(TIME = 3310 msec)

(ALL THE SHEAR EVOLUTION EQUATIONS ARE FULFILLED IDENTICALLY)
(SHEAR EVOLUTION EQUATIONS calculated)
(TIME = 3820 msec)

>   magweyl    = - (1/2) C
            2 3

(magweyl calculated)
(TIME = 3890 msec)

(ALL THE MAGNETIC CONSTRAINTS ARE FULFILLED IDENTICALLY)
(magcons calculated)
(TIME = 5170 msec)

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

END OF WORK
(RUN TIME = 5170 msec)
```

E.8 Example VIII: Application of the program "curvature".

In this example, the program will calculate the curvature tensor for a 3-dimensional manifold of constant positive curvature (in fact, a 3-sphere; the connection coefficients used here as input data were previously calculated by another program of the Ortocartan set as the Christoffel symbols for the metric of a 3-sphere). This example is so simple in order that the readers can look up the answer in textbooks and verify that it is correct. The number of dimensions of the manifold, n, can be arbitrary and it is the first argument of the function "curvature". The connection coefficients are assumed symmetric, and so there should be $\frac{1}{2}n^2(n+1)$ of them. The program checks whether the number of the connection coefficients actually given is equal to this. The order in which the conection coefficients should be given is described in Appendix C.2, it is the obvious one.

The input data is here:

```
(setq !*lower nil)
(curvature 3 '(
```

```
   (the curvature of christoffels of spherical space)
   (coordinates r th ph)
   (connection 0 0 0 (- (sin r) * (cos r)) 0 (- (sin r)*(cos r)*(sin th) ^ 2)
               0 ((cos r) / (sin r)) 0 0 0 (- (cos th) * (sin th))
               0 0 ((cos r) / (sin r)) 0 ((cos th) / (sin th)) 0
                       )
           ))
 (setq !*lower t)
```

and the results are:

```
(the curvature of christoffels of spherical space)

              0
>   connection      = - cos (r) sin (r)
                1 1

              0                             2
>   connection      = - cos (r) sin (r) sin (th)
                2 2

              1                   -1
>   connection      = cos (r) sin  (r)
                0 1

              1
>   connection      = - cos (th) sin (th)
                2 2

              2                   -1
>   connection      = cos (r) sin  (r)
                0 2

              2                    -1
>   connection      = cos (th) sin  (th)
                1 2

(CONNECTION COEFFICIENTS completed)
(TIME = 60 msec)


              0            2
>   ncurvature        = sin (r)
                1 0 1
```

```
              0            2       2
>   ncurvature        = sin (r) sin (th)
                2 0 2

              1
>   ncurvature        = -1
                0 0 1

              1            2       2
>   ncurvature        = sin (r) sin (th)
                2 1 2

              2
>   ncurvature        = -1
                0 0 2

              2             2
>   ncurvature        = - sin (r)
                1 1 2

(ncurvature calculated)
(TIME = 190 msec)

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

END OF WORK
(RUN TIME = 190 msec)
```

E.9 Example IX: Application of the program "landlagr".

The Landau-Lifshitz lagrangian equals $\sqrt{-g}\mathcal{R}$, where g is the determinant of the metric tensor and $-\mathcal{R}$ is the Ricci scalar with the derivatives of the Christoffel symbols dropped. The example will be a diagonal Bianchi type I metric, for which the reduced lagrangian is known to provide the correct Einstein equations.

The input data is:

```
(landlagr '(
    (lagrangian for a diagonal Bianchi I metric)
    (coordinates t x y z)
    (functions f1(t) f2(t) f3(t))
    (ematrix 1 0 0 0 0 f1 0 0 0 0
             f2 0 0 0 0 f3)
```

```
    (rmargin 61)
            ))
```

and the result is:

```
(lagrangian for a diagonal bianchi i metric)

           0
>   ematrix  . = 1
            0

           1
>   ematrix  . = f1
            1

           2
>   ematrix  . = f2
            2

           3
>   ematrix  . = f3
            3

(ematrix completed)
(TIME = 20 msec)


>   DETERMINANT EMATRIX = f1 f2 f3


(DETERMINANT EMATRIX calculated)
(TIME = 30 msec)


      0
>   ie.   = 1
        0

      1       -1
>   ie.   = f1
        1

      2       -1
>   ie.   = f2
        2
```

```
      3       -1
>   ie.   = f3
        3

(ie calculated)
(TIME = 220 msec)



>   metric    = 1
          0 0


                    2
>   metric    = - f1
          1 1


                    2
>   metric    = - f2
          2 2


                    2
>   metric    = - f3
          3 3


(metric calculated)
(TIME = 250 msec)



             0 0
>   invmetric    = 1


             1 1       -2
>   invmetric    = - f1



             2 2       -2
>   invmetric    = - f2
```

```
             3 3       -2
>   invmetric    = - f3


(invmetric calculated)
(TIME = 310 msec)


          1               -1
>   agamma      = (1/2) f1   f1,
             0 1                 t

          2               -1
>   agamma      = (1/2) f2   f2,
             0 2                 t

          3               -1
>   agamma      = (1/2) f3   f3,
             0 3                 t

(agamma calculated)
(TIME = 520 msec)

(agamma completed)
(TIME = 540 msec)

         0         -1
>   gamma      = f1   f1,
            1 1           t


         0         -1
>   gamma      = f2   f2,
            2 2           t


         0         -1
>   gamma      = f3   f3,
            3 3           t


(gamma calculated)
```

```
(TIME = 590 msec)

(gamma completed)
(TIME = 590 msec)

               0
>   christoffel      = f1 f1,
                 1 1         t


               0
>   christoffel      = f2 f2,
                 2 2         t


               0
>   christoffel      = f3 f3,
                 3 3         t


               1         -1
>   christoffel      = f1   f1,
                 0 1           t


               2         -1
>   christoffel      = f2   f2,
                 0 2           t


               3         -1
>   christoffel      = f3   f3,
                 0 3           t


(CHRISTOFFEL SYMBOLS calculated)
(TIME = 710 msec)

(CHRISTOFFEL SYMBOLS completed)
(TIME = 720 msec)


>   landlagr = 2 f1 f2,   f3,   + 2 f2 f1,   f3,   + 2 f3 f1
                       t     t            t     t
```

```
>       ,   f2,
         t     t


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

END OF WORK
(RUN TIME = 870 msec)
```

This lagrangian can then be used as data for the program "eulagr", and the resulting Euler-Lagrange equations can be compared with the Einstein equations derived for the same metric in the ordinary way.

E.10 Example X: Application of the program "eulagr".

The program will derive the Newtonian equations of motion for a point particle of mass m in the cartesian coordinates $\{x, y, z\}$ from the lagrangian

$$L = \frac{1}{2}m(\dot{x}^2 + \dot{y}^2 + \dot{z}^2) - V(x, y, z),$$

where V is a potential and $x(t), y(t), z(t)$ are the equations of a trajectory of the particle. The input data are:

```
(setq !*lower nil)
(eulagr '(
   (The lagrangian for the Newtonian equations of motion
    in 3 dimensions)
   (constants m)
   (parameter t)
   (functions x(t) y(t) z(t) V(x y z) )
   (variables x y z)
   (lagrangian ((1 2) * m * ((der t x) ^ 2 + (der t y) ^ 2
      + (der t z) ^ 2) - V))
        ))
(setq !*lower t)
```

and the results are:

```
(The lagrangian for the Newtonian equations of motion in 3 dimensions)

                                    2               2               2
>   lagrangian = - V + (1/2) m x,    + (1/2) m y,    + (1/2) m z,
                                  t               t               t
```

```
(THIS IS THE VARIATIONAL DERIVATIVE BY x)

>   eulagr  = m x,     + V,
          0       t t      x


(THIS IS THE VARIATIONAL DERIVATIVE BY y)

>   eulagr  = m y,     + V,
          1       t t      y


(THIS IS THE VARIATIONAL DERIVATIVE BY z)

>   eulagr  = m z,     + V,
          2       t t      z

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

END OF WORK
(RUN TIME = 100 msec)
```

E.11 Example XI: Application of the program "squint".

In order to make the result easy to verify, we shall use the program "squint" to find a first integral of the equations found in the previous example. We shall at first pretend that we do not know what the integral should be and will assume that it is a general polynomial of second degree in the first derivatives by t of the functions $x(t)$, $y(t)$ and $z(t)$. Note how the (markers ...) were used to simplify the substitutions: the single equation (der t t M) = (der M V) represents the 3 equations $d^2x^i/dt^2 = \partial V/\partial x^i$ for $i = 1, 2, 3$ simultaneously.

The input data are:

```
(setq !*lower nil)
(squint'(
   (a first integral of the Newtonian equations of motion)
   (constants m)
   (parameter t)
   (functions x(t) y(t) z(t) V(x y z) Q11(x y z) Q12(x y z) Q13(x y z) Q22(x y z)
   Q23(x y z) Q33(x y z) L1(x y z) L2(x y z) L3(x y z) E(x y z) )
   (variables x y z)
(integral (Q11 * (der t x) ^ 2 + 2 * Q12 * (der t x) * (der t y)
    + 2 * Q13 * (der t x) * (der t z) + Q22 * (der t y) ^ 2
```

```
    + 2 * Q23 * (der t y) * (der t z) + Q33 * (der t z) ^ 2
    + L1 * (der t x) + L2 * (der t y) + L3 * (der t z) + E) )
(markers M)
   (substitutions
    maineq
(der t t M) = (- (der M V) / m)
     )
(dont print maineq)
         ))
(setq !*lower t)
```

and the results are:

```
(a first integral of the Newtonian equations of motion)


                           2
>   integral = E + Q11 x,    + 2 Q12 x,   y,   + 2 Q13 x,   z,   + Q22
                         t            t    t           t    t

            2                               2
>       y,    + 2 Q23 y,   z,   + Q33 z,    + L1 x,   + L2 y,   + L3 z,
          t            t    t          t          t         t


>
        t


substitutions

maineq

                -1
>   M,     = - m   V,
      t t           M


                                    3
>   THIS IS THE COEFFICIENT OF   x,
                                   t

>   equation  = Q11,
            1       x
```

```
                                    2
>   THIS IS THE COEFFICIENT OF   x,    y,
                                   t     t

>   equation  = 2 Q12,   + Q11,
           2         x        y

                                    2
>   THIS IS THE COEFFICIENT OF   x,    z,
                                   t     t

>   equation  = 2 Q13,   + Q11,
           3         x        z

                                      2
>   THIS IS THE COEFFICIENT OF   x,   y,
                                   t    t

>   equation  = Q22,   + 2 Q12,
           4       x          y

>   THIS IS THE COEFFICIENT OF   x,   y,   z,
                                   t    t    t

>   equation  = 2 Q23,   + 2 Q13,   + 2 Q12,
           5         x          y          z

                                      2
>   THIS IS THE COEFFICIENT OF   x,   z,
                                   t    t

>   equation  = Q33,   + 2 Q13,
           6       x          z

                                    3
>   THIS IS THE COEFFICIENT OF   y,
                                   t

>   equation  = Q22,
           7       y

                                    2
>   THIS IS THE COEFFICIENT OF   y,    z,
                                   t     t
```

```
>   equation  = 2 Q23,   + Q22,
                8         y        z

                                                2
>   THIS IS THE COEFFICIENT OF   y,   z,
                                   t    t

>   equation  = Q33,   + 2 Q23,
                9       y          z

                                     3
>   THIS IS THE COEFFICIENT OF   z,
                                   t

>   equation   = Q33,
                10       z

                                   2
>   THIS IS THE COEFFICIENT OF   x,
                                   t

>   equation   = L1,
                11      x

>   THIS IS THE COEFFICIENT OF   x,   y,
                                   t    t

>   equation   = L2,   + L1,
                12      x       y

>   THIS IS THE COEFFICIENT OF   x,   z,
                                   t    t

>   equation   = L3,   + L1,
                13      x       z

                                     2
>   THIS IS THE COEFFICIENT OF   y,
                                   t

>   equation   = L2,
                14      y
```

```
>   THIS IS THE COEFFICIENT OF   y,   z,
                                  t    t


>   equation   = L3,   + L2,
           15      y       z

                                    2
>   THIS IS THE COEFFICIENT OF   z,
                                  t

>   equation   = L3,
           16      z

>   THIS IS THE COEFFICIENT OF   x,
                                  t

                     -1               -1               -1
>   equation   = - 2 m   Q11 V,   - 2 m   Q12 V,   - 2 m   Q13 V,   + E
           17                x                y                z

>       ,
        x

>   THIS IS THE COEFFICIENT OF   y,
                                  t

                     -1               -1               -1
>   equation   = - 2 m   Q12 V,   - 2 m   Q22 V,   - 2 m   Q23 V,   + E
           18                x                y                z

>       ,
        y

>   THIS IS THE COEFFICIENT OF   z,
                                  t

                     -1               -1               -1
>   equation   = - 2 m   Q13 V,   - 2 m   Q23 V,   - 2 m   Q33 V,   + E
           19                x                y                z

>       ,
        z

>   THESE ARE THE TERMS THAT ARE   FREE OF THE DERIVATIVES
```

```
                    -1            -1            -1
>   equation   = - m   L1 V,   - m   L2 V,   - m   L3 V,
           20              x             y             z


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

END OF WORK
(RUN TIME = 820 msec)
```

Now we shall substitute the well-known solution of these equations into the data and see what happens.

```
(setq !*lower nil)
(squint'(
   (a first integral of the Newtonian equations of motion - the final result)
   (constants m)
   (parameter t)
   (functions x(t) y(t) z(t) V(x y z))
   (variables x y z)
(integral ((1 2) * m * ((der t x) ^ 2 + (der t y) ^ 2 + (der t z) ^ 2) + V) )
(markers M)
   (substitutions
    maineq
(der t t M) = (- (der M V) / m)
     )
     (dont print maineq)
          ))
(setq !*lower t)
```

The result is:

```
(a first integral of the Newtonian equations of motion - the final result)

                                2               2               2
>   integral = V + (1/2) m x,    + (1/2) m y,    + (1/2) m z,
                              t               t               t

substitutions

maineq
```

```
                -1
>   M,     = - m   V,
      t t            M


(THE FIRST INTEGRAL IS ALREADY MAXIMALLY SIMPLIFIED
AND IS EXPLICITLY CONSTANT)
>   maineq = 0



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

END OF WORK
(RUN TIME = 130 msec)
```

****

You can produce many more examples by yourself if you use the sets of input data recorded in the *.tes-files on the Ortocartan distribution diskette. Do not rewrite them, but use the editor to cut out single calls to Ortocartan or to the other functions. Good luck and enjoy it!

***** THIS IS THE END OF THE MANUAL *****