EXCALC: A System for Doing Calculations in the Calculus of Modern Differential Geometry

Eberhard Schrüfer
Institute SCAI.Alg
German National Research Center for Information Technology (GMD)
Schloss Birlinghoven
D-53754 Sankt Augustin
Germany
Email: schruefer@gmd.de

May 3, 1999

Acknowledgments

This program was developed over several years. I would like to express my deep gratitude to Dr. Anthony Hearn for his continuous interest in this work, and especially for his hospitality and support during a visit in 1984/85 at the RAND Corporation, where substantial progress on this package could be achieved. The Heinrich Hertz-Stiftung supported this visit. Many thanks are also due to Drs. F.W. Hehl, University of Cologne, and J.D. McCrea, University College Dublin, for their suggestions and work on testing this program.

1 Introduction

EXCALC is designed for easy use by all who are familiar with the calculus of Modern Differential Geometry. Its syntax is kept as close as possible to standard textbook notations. Therefore, no great experience in writing computer algebra programs is required. It is almost possible to input to the computer the same as what would have been written down for a hand-

calculation. For example, the statement

```
f*x^y + u _| (y^z^x)
```

would be recognized by the program as a formula involving exterior products and an inner product. The program is currently able to handle scalar-valued exterior forms, vectors and operations between them, as well as non-scalar valued forms (indexed forms). With this, it should be an ideal tool for studying differential equations, doing calculations in general relativity and field theories, or doing such simple things as calculating the Laplacian of a tensor field for an arbitrary given frame. With the increasing popularity of this calculus, this program should have an application in almost any field of physics and mathematics.

Since the program is completely embedded in REDUCE, all features and facilities of REDUCE are available in a calculation. Even for those who are not quite comfortable in this calculus, there is a good chance of learning it by just playing with the program.

This is the last release of version 2. A much extended differential geometry package (which includes complete symbolic index simplification, tensors, mappings, bundles and others) is under development.

Complaints and comments are appreciated and should be send to the author. If the use of this program leads to a publication, this document should be cited, and a copy of the article to the above address would be welcome.

2 Declarations

Geometrical objects like exterior forms or vectors are introduced to the system by declaration commands. The declarations can appear anywhere in a program, but must, of course, be made prior to the use of the object. Everything that has no declaration is treated as a constant; therefore zero-forms must also be declared.

An exterior form is introduced by

PFORM <*declaration*$_1$>, <*declaration*$_2$>, ...;

where

<*declaration*> ::= <*name*> | <*list of names*>=<*number*> | <*identifier*> |

<*expression*>
<*name*> ::= <*identifier*> | <*identifier*>(<*arguments*>)

For example

```
pform u=k,v=4,f=0,w=dim-1;
```

declares `U` to be an exterior form of degree `K`, `V` to be a form of degree 4, `F` to be a form of degree 0 (a function), and `W` to be a form of degree `DIM`-1.

If the exterior form should have indices, the declaration would be

```
pform curv(a,b)=2,chris(a,b)=1;
```

The names of the indices are arbitrary.

Exterior forms of the same degree can be grouped into lists to save typing.

```
pform {x,y,z}=0,{rho(k,l),u,v(k)}=1;
```

The declaration of vectors is similar. The command `TVECTOR` takes a list of names.

TVECTOR <*name*$_1$>, <*name*$_2$>, ...;

For example, to declare `X` as a vector and `COMM` as a vector with two indices, one would say

```
tvector x,comm(a,b);
```

If a declaration of an already existing name is made, the old declaration is removed, and the new one is taken.

The exterior degree of a symbol or a general expression can be obtained with the function

EXDEGREE <*expression*>;

Example:

```
exdegree(u + 3*chris(k,-k));

1
```

3 Exterior Multiplication

Exterior multiplication between exterior forms is carried out with the nary infix operator ^ (wedge). Factors are ordered according to the usual ordering in REDUCE using the commutation rule for exterior products.

Example 1

```
pform u=1,v=1,w=k;

u^v;

U^V

v^u;

- U^V

u^u;

0

w^u^v;

      K
( - 1) *U^V^W

(3*u-a*w)^(w+5*v)^u;

A*(5*U^V^W - U^W^W)
```

It is possible to declare the dimension of the underlying space by

SPACEDIM <*number*> | <*identifier*>;

If an exterior product has a degree higher than the dimension of the space, it is replaced by 0:

```
spacedim 4;

pform u=2,v=3;
```

```
u^v;

0
```

4 Partial Differentiation

Partial differentiation is denoted by the operator `@`. Its capability is the same as the REDUCE `DF` operator.

Example 2

```
@(sin x,x);

COS(X)

@(f,x);

0
```

An identifier can be declared to be a function of certain variables. This is done with the command `FDOMAIN`. The following would tell the partial differentiation operator that `F` is a function of the variables `X` and `Y` and that `H` is a function of `X`.

```
fdomain f=f(x,y),h=h(x);
```

Applying `@` to `F` and `H` would result in

```
@(x*f,x);

F + X*@  F
       X

@(h,y);

0
```

The partial derivative symbol can also be an operator with a single argument. It then represents a natural base element of a tangent vector.

Example 3

```
a*@ x + b*@ y;

A*@  + B*@
   X      Y
```

5 Exterior Differentiation

Exterior differentiation of exterior forms is carried out by the operator `d`. Products are normally differentiated out, *i.e.*

```
pform x=0,y=k,z=m;

d(x * y);

X*d Y + d X^Y

d(r*y);

R*d Y

d(x*y^z);

      K
( - 1) *X*Y^d Z  + X*d Y^Z + d X^Y^Z
```

This expansion can be suppressed by the command `NOXPND D`.

```
noxpnd d;

d(y^z);

d(Y^Z)
```

To obtain a canonical form for an exterior product when the expansion is switched off, the operator `D` is shifted to the right if it appears in the leftmost place.

```
d y ^ z;

        K
```

```
- ( - 1) *Y^d Z + d(Y^Z)
```

Expansion is performed again when the command `XPND D` is executed.

Functions which are implicitly defined by the `FDOMAIN` command are expanded into partial derivatives:

```
pform x=0,y=0,z=0,f=0;

fdomain f=f(x,y);

d f;

@  F*d X + @  F*d Y
 X          Y
```

If an argument of an implicitly defined function has further dependencies the chain rule will be applied *e.g.*

```
fdomain y=y(z);

d f;

@  F*d X + @  F*@  Y*d Z
 X          Y    Z
```

Expansion into partial derivatives can be inhibited by `NOXPND @` and enabled again by `XPND @`.

The operator is of course aware of the rules that a repeated application always leads to zero and that there is no exterior form of higher degree than the dimension of the space.

```
d d x;

0

pform u=k;
spacedim k;

d u;

0
```

6 Inner Product

The inner product between a vector and an exterior form is represented by the diphthong _| (underscore or-bar), which is the notation of many textbooks. If the exterior form is an exterior product, the inner product is carried through any factor.

Example 4

```
pform x=0,y=k,z=m;

tvector u,v;

u _| (x*y^z);

          K
X*(( - 1) *Y^U _| Z + U _| Y^Z)
```

In repeated applications of the inner product to the same exterior form the vector arguments are ordered *e.g.*

```
(u+x*v) _| (u _| (3*z));

- 3*U _| V _| Z
```

The duality of natural base elements is also known by the system, *i.e.*

```
pform {x,y}=0;

(a*@ x+b*@(y)) _| (3*d x-d y);

3*A - B
```

7 Lie Derivative

The Lie derivative can be taken between a vector and an exterior form or between two vectors. It is represented by the infix operator |_ . In the case of Lie differentiating, an exterior form by a vector, the Lie derivative is expressed through inner products and exterior differentiations, *i.e.*

```
pform z=k;
```

```
tvector u;

u |_ z;

U _| d Z + d(U _| Z)
```

If the arguments of the Lie derivative are vectors, the vectors are ordered using the anticommutivity property, and functions (zero forms) are differentiated out.

Example 5

```
tvector u,v;

v |_ u;

- U |_ V

pform x=0,y=0;

(x*u) |_ (y*v);

- U*Y*V _| d X + V*X*U _| d Y + X*Y*U |_ V
```

8 Hodge-* Duality Operator

The Hodge-* duality operator maps an exterior form of degree `K` to an exterior form of degree `N-K`, where `N` is the dimension of the space. The double application of the operator must lead back to the original exterior form up to a factor. The following example shows how the factor is chosen here

```
spacedim n;
pform x=k;

# # x;

        2
      (K  + K*N)
```

```
( - 1)          *X*SGN
```

The indeterminate SGN in the above example denotes the sign of the determinant of the metric. It can be assigned a value or will be automatically set if more of the metric structure is specified (via COFRAME), *i.e.* it is then set to $g/|g|$, where g is the determinant of the metric. If the Hodge-* operator appears in an exterior product of maximal degree as the leftmost factor, the Hodge-* is shifted to the right according to

```
pform {x,y}=k;

# x ^ y;

         2
       (K  + K*N)
( - 1)           *X^# Y
```

More simplifications are performed if a coframe is defined.

9 Variational Derivative

The function `VARDF` returns as its value the variation of a given Lagrangian n-form with respect to a specified exterior form (a field of the Lagrangian). In the shared variable `BNDEQ!*`, the expression is stored that has to yield zero if integrated over the boundary.

Syntax:

VARDF(<*Lagrangian n-form*>,<*exterior form*>)

Example 6

```
spacedim 4;

pform l=4,a=1,j=3;

l:=-1/2*d a ^ # d a - a^# j$  %Lagrangian of the e.m. field

vardf(l,a);

- (# J + d # d A)             %Maxwell's equations

bndeq!*;
```

```
- 'A^# d A                          %Equation at the boundary
```

Restrictions:

In the current implementation, the Lagrangian must be built up by the fields and the operations `d`, `#`, and `@`. Variation with respect to indexed quantities is currently not allowed.

For the calculation of the conserved currents induced by symmetry operators (vector fields), the function `NOETHER` is provided. It has the syntax:

NOETHER(<*Lagrangian n-form*>,<*field*>,<*symmetry generator*>)

Example 7

```
pform l=4,a=1,f=2;

spacedim 4;

l:= -1/2*d a^#d a;   %Free Maxwell field;

tvector x(k);        %An unspecified generator;

noether(l,a,x(-k));

( - 2*d(X _|A)^# d A - (X _|d A)^# d A + d A^(X _|# d A))/2
         K               K                      K
```

The above expression would be the canonical energy momentum 3-forms of the Maxwell field, if X is interpreted as a translation;

10 Handling of Indices

Exterior forms and vectors may have indices. On input, the indices are given as arguments of the object. A positive argument denotes a superscript and a negative argument a subscript. On output, the indexed quantity is displayed two dimensionally if `NAT` is on. Indices may be identifiers or numbers.

Example 8

```
pform om(k,l)=m,e(k)=1;
```

```
e(k)^e(-l);

 K
E ^E
    L

om(4,-2);

  4
OM
   2
```

In the current release, full simplification is performed only if an index range is specified. It is hoped that this restriction can be removed soon. If the index range (the values that the indices can obtain) is specified, the given expression is evaluated for all possible index values, and the summation convention is understood.

Example 9

```
indexrange t,r,ph,z;

pform e(k)=1,s(k,l)=2;

w := e(k)*e(-k);

         T       R        PH       Z
W := E *E  + E *E  + E  *E   + E *E
      T       R       PH        Z


s(k,l):=e(k)^e(l);

 T T
S    := 0

 R T       T  R
S    := - E ^E
```

```
 PH T       T  PH
S     := - E ^E

  .
  .
  .
```

If the expression to be evaluated is not an assignment, the values of the expression are displayed as an assignment to an indexed variable with name `NS`. This is done only on output, *i.e.* no actual binding to the variable NS occurs.

```
e(k)^e(l);

  T T
NS    := 0

  R T       T  R
NS    := - E ^E
  .
  .
  .
```

It should be noted, however, that the index positions on the variable NS can sometimes not be uniquely determined by the system (because of possible reorderings in the expression). Generally it is advisable to use assignments to display complicated expressions.

A range can also be assigned to individual index-names. For example, the declaration

```
indexrange {k,l}={x,y,z},{u,v,w}={1,2};
```

would assign to the index identifiers k,l the range values x,y,z and to the index identifiers u,v,w the range values 1,2. The use of an index identifier not listed in previous indexrange statements has the range of the union of all given index ranges.

With the above example of an indexrange statement, the following index evaluations would take place

```
pform w n=0;
```

```
w(k)*w(-k);

    X        Y        Z
W *W  + W *W  + W *W
 X        Y        Z

w(u)*w(-u);

    1        2
W *W  + W *W
 1        2

w(r)*w(-r);

    1        2        X        Y        Z
W *W  + W *W  + W *W  + W *W  + W *W
 1        2        X        Y        Z
```

In certain cases, one would like to inhibit the summation over specified index names, or at all. For this the command

NOSUM <*indexname*$_1$>, ...;

and the switch `NOSUM` are available. The command `NOSUM` has the effect that summation is not performed over those indices which had been listed. The command `RENOSUM` enables summation again. The switch `NOSUM`, if on, inhibits any summation.

It is possible to declare symmetry properties for an indexed quantity by the command `INDEX_SYMMETRIES`. A prototypical example is as follows

```
index_symmetries u(k,l,m,n): symmetric     in {k,l},{m,n}
                             antisymmetric in {{k,l},{m,n}},
                 g(k,l),h(k,l): symmetric;
```

It declares the object `u` symmetric in the first two and last two indices and antisymmetric with respect to commutation of the given index pairs. If an object is completely symmetric or antisymmetric, the indices need not to be given after the corresponding keyword as shown above for `g` and `h`.

If applicable, this command should be issued, since great savings in memory and execution time result. Only strict components are printed.

The commands symmetric and antisymmetric of earlier releases have no effect.

11 Metric Structures

A metric structure is defined in **EXCALC** by specifying a set of basis one-forms (the coframe) together with the metric.

Syntax:

COFRAME <*identifier*><*(index$_1$)*>=<*expression$_1$*>,
<*identifier*><*(index$_2$)*>=<*expression$_2$*>,
.
.
.
<*identifier*><*(index$_n$)*>=<*expression$_n$*>
WITH METRIC <*name*>=<*expression*>;

This statement automatically sets the dimension of the space and the index range. The clause `WITH METRIC` can be omitted if the metric is Euclidean and the shorthand `WITH SIGNATURE` <*`diagonal elements`*> can be used in the case of a pseudo-Euclidean metric. The splitting of a metric structure in its metric tensor coefficients and basis one-forms is completely arbitrary including the extremes of an orthonormal frame and a coordinate frame.

Example 10

```
coframe e r=d r, e(ph)=r*d ph
  with metric g=e(r)*e(r)+e(ph)*e(ph);    %Polar coframe

coframe e(r)=d r,e(ph)=r*d(ph);           %Same as before

coframe o(t)=d t, o x=d x
  with signature -1,1;                    %A Lorentz coframe

coframe b(xi)=d xi, b(eta)=d eta          %A lightcone coframe
  with metric w=-1/2*(b(xi)*b(eta)+b(eta)*b(xi));
```

```
coframe e r=d r, e ph=d ph                %Polar coordinate
  with metric g=e r*e r+r**2*e ph*e ph;   %basis
```

Individual elements of the metric can be accessed just by calling them with the desired indices. The value of the determinant of the covariant metric is stored in the variable `DETM!*`. The metric is not needed for lowering or raising of indices as the system performs this automatically, *i.e.* no matter in what index position values were assigned to an indexed quantity, the values can be retrieved for any index position just by writing the indexed quantity with the desired indices.

Example 11

```
coframe e t=d t,e x=d x,e y=d y
 with signature -1,1,1;

pform f(k,l)=0;

antisymmetric f;

f(-t,-x):=ex$ f(-x,-y):=b$  f(-t,-y):=0$
on nero;

f(k,-l):=f(k,-l);

 X
F    := - EX
   T

 T
F    := - EX
   X

 Y
F    := - B
   X

 X
```

```
F    := B
   Y
```

Any expression containing differentials of the coordinate functions will be transformed into an expression of the basis one-forms.The system also knows how to take the exterior derivative of the basis one-forms.

Example 12(Spherical coordinates)

```
coframe e(r)=d(r), e(th)=r*d(th), e(ph)=r*sin(th)*d(ph);

d r^d th;

  R  TH
(E ^E  )/R

d(e(th));

  R  TH
(E ^E  )/R


pform f=0;

fdomain f=f(r,th,ph);

factor e;

on rat;

d f;       %The "gradient" of F in spherical coordinates;

 R          TH              PH
E *@  F + (E  *@   F)/R + (E  *@   F)/(R*SIN(TH))
    R           TH              PH
```

The frame dual to the frame defined by the `COFRAME` command can be introduced by **FRAME** command.

FRAME <*identifier*>;

This command causes the dual property to be recognized, and the tangent

vectors of the coordinate functions are replaced by the frame basis vectors.

Example 13

```
coframe b r=d r,b ph=r*d ph,e z=d z; %Cylindrical coframe;

frame x;

on nero;

x(-k) _| b(l);

    R
NS    := 1
  R

     PH
NS      := 1
  PH

    Z
NS    := 1
  Z

x(-k) |_ x(-l);       %The commutator of the dual frame;


NS     := X  /R
  PH R     PH


NS     := ( - X  )/R  %i.e. it is not a coordinate base;
  R PH         PH
```

As a convenience, the frames can be displayed at any point in a program by the command `DISPLAYFRAME;`.

The Hodge-* duality operator returns the explicitly constructed dual element if applied to coframe base elements. The metric is properly taken into account.

The total antisymmetric Levi-Cevita tensor `EPS` is also available. The value of `EPS` with an even permutation of the indices in a covariant position is taken to be +1.

12 Riemannian Connections

The command `RIEMANNCONX` is provided for calculating the connection 1 forms. The values are stored on the name given to `RIEMANNCONX`. This command is far more efficient than calculating the connection from the differential of the basis one-forms and using inner products.

Example 14(Calculate the connection 1-form and curvature 2-form on S(2))

```
coframe e th=r*d th,e ph=r*sin(th)*d ph;

riemannconx om;

om(k,-l);                %Display the connection forms;

  TH
NS      := 0
     TH

  PH         PH
NS      := (E  *COS(TH))/(SIN(TH)*R)
     TH

  TH            PH
NS      := ( - E  *COS(TH))/(SIN(TH)*R)
     PH

  PH
NS      := 0
     PH

pform curv(k,l)=2;
```

```
curv(k,-l):=d om(k,-l) + om(k,-m)^om(m-l);
              %The curvature forms

    TH
CURV      := 0
       TH

    PH             TH  PH   2
CURV      := ( - E  ^E  )/R
       TH               %Of course it was a sphere with
                        %radius R.

    TH          TH  PH   2
CURV      := (E  ^E  )/R
       PH

    PH
CURV      := 0
       PH
```

13 Ordering and Structuring

The ordering of an exterior form or vector can be changed by the command `FORDER`. In an expression, the first identifier or kernel in the arguments of `FORDER` is ordered ahead of the second, and so on, and ordered ahead of all not appearing as arguments. This ordering is done on the internal level and not only on output. The execution of this statement can therefore have tremendous effects on computation time and memory requirements. `REMFORDER` brings back standard ordering for those elements that are listed as arguments.

An expression can be put in a more structured form by renaming a subexpression. This is done with the command `KEEP` which has the syntax

KEEP <*name*$_1$>=<*expression*$_1$>,<*name*$_2$>=<*expression*$_2$>, ...

The effect is that rules are set up for simplifying <*name*> without introducing its definition in an expression. In an expression the system also tries by reordering to generate as many instances of <*name*> as possible.

Example 15

```
pform x=0,y=0,z=0,f=0,j=3;

keep j=d x^d y^d z;

j;

J

d j;

0

j^d x;

0

fdomain f=f(x);

d f^d y^d z;

@  F*J
 X
```

The capabilities of KEEP are currently very limited. Only exterior products should occur as righthand sides in KEEP.

14 Summary of Operators and Commands

Table 1 summarizes EXCALC commands and the page number they are defined on.

^	Exterior Multiplication	4
@	Partial Differentiation	5
@	Tangent Vector	5
#	Hodge-* Operator	9
_|	Inner Product	8
|_	Lie Derivative	8
COFRAME	Declaration of a coframe	16
d	Exterior differentiation	6
DISPLAYFRAME	Displays the frame	19
EPS	Levi-Civita tensor	20
EXDEGREE	Calculates the exterior degree of an expression	3
FDOMAIN	Declaration of implicit dependencies	5
FORDER	Ordering command	21
FRAME	Declares the frame dual to the coframe	18
INDEXRANGE	Declaration of indices	13
INDEX_SYMMETRIES	Declares arbitrary index symmetry properties	15
KEEP	Structuring command	21
METRIC	Clause of COFRAME to specify a metric	16
NOETHER	Calculates the Noether current	12
NOSUM	Inhibits summation convention	15
NOXPND d	Inhibits the use of product rule for d	6
NOXPND @	Inhibits expansion into partial derivatives	7
PFORM	Declaration of exterior forms	2
REMFORDER	Clears ordering	21
RENOSUM	Enables summation convention	15
RIEMANNCONX	Calculation of a Riemannian Connection	20
SIGNATURE	Clause of COFRAME to specify a pseudo-Euclidean metric	16
SPACEDIM	Command to set the dimension of a space	4
TVECTOR	Declaration of vectors	3
VARDF	Variational derivative	11
XPND d	Enables the use of product rule for d (default)	7
XPND @	Enables expansion into partial derivatives (default)	7

Table 1: EXCALC Command Summary

15 Examples

The following examples should illustrate the use of **EXCALC**. It is not intended to show the most efficient or most elegant way of stating the problems; rather the variety of syntactic constructs are exemplified. The examples are on a test file distributed with **EXCALC**.

```
% Problem: Calculate the PDE's for the isovector of the heat equation.
% --------
%         (c.f. B.K. Harrison, f.B. Estabrook, "Geometric Approach...",
%          J. Math. Phys. 12, 653, 1971)

% The heat equation @   psi = @  psi is equivalent to the set of exterior
%                    xx        t

% equations (with u=@ psi, y=@ psi):
%                    T        x


pform {psi,u,x,y,t}=0,a=1,{da,b}=2;

a := d psi - u*d t - y*d x;

da := - d u^d t - d y^d x;

b := u*d x^d t - d y^d t;


% Now calculate the PDE's for the isovector.

tvector v;

pform {vpsi,vt,vu,vx,vy}=0;
fdomain vpsi=vpsi(psi,t,u,x,y),vt=vt(psi,t,u,x,y),vu=vu(psi,t,u,x,y),
                               vx=vx(psi,t,u,x,y),vy=vy(psi,t,u,x,y);

v := vpsi*@ psi + vt*@ t + vu*@ u + vx*@ x + vy*@ y;


factor d;
on rat;

i1 := v |_ a - l*a;
```

```
pform o=1;

o := ot*d t + ox*d x + ou*d u + oy*d y;

fdomain f=f(psi,t,u,x,y);

i11 := v _| d a - l*a + d f;

let vx=-@(f,y),vt=-@(f,u),vu=@(f,t)+u*@(f,psi),vy=@(f,x)+y*@(f,psi),
    vpsi=f-u*@(f,u)-y*@(f,y);

factor ^;

i2 := v |_ b - xi*b - o^a + zeta*da;

let ou=0,oy=@(f,u,psi),ox=-u*@(f,u,psi),
    ot=@(f,x,psi)+u*@(f,y,psi)+y*@(f,psi,psi);

i2;

let zeta=-@(f,u,x)-@(f,u,y)*u-@(f,u,psi)*y;

i2;

let xi=-@(f,t,u)-u*@(f,u,psi)+@(f,x,y)+u*@(f,y,y)+y*@(f,y,psi)+@(f,psi);

i2;

let @(f,u,u)=0;

i2;      % These PDE's have to be solved.


clear a,da,b,v,i1,i11,o,i2,xi,t;
remfdomain f,vpsi,vt,vu,vx,vy;
clear @(f,u,u);


% Problem:
% --------
% Calculate the integrability conditions for the system of PDE's:
% (c.f. B.F. Schutz, "Geometrical Methods of Mathematical Physics"
% Cambridge University Press, 1984, p. 156)
```

```
% @ z /@ x + a1*z  + b1*z  = c1
%    1           1       2

% @ z /@ y + a2*z  + b2*z  = c2
%    1           1       2

% @ z /@ x + f1*z  + g1*z  = h1
%    2           1       2

% @ z /@ y + f2*z  + g2*z  = h2
%    2           1       2      ;


pform w(k)=1,integ(k)=4,{z(k),x,y}=0,{a,b,c,f,g,h}=1,
      {a1,a2,b1,b2,c1,c2,f1,f2,g1,g2,h1,h2}=0;

fdomain  a1=a1(x,y),a2=a2(x,y),b1=b1(x,y),b2=b2(x,y),
         c1=c1(x,y),c2=c2(x,y),f1=f1(x,y),f2=f2(x,y),
         g1=g1(x,y),g2=g2(x,y),h1=h1(x,y),h2=h2(x,y);


a:=a1*d x+a2*d y$
b:=b1*d x+b2*d y$
c:=c1*d x+c2*d y$
f:=f1*d x+f2*d y$
g:=g1*d x+g2*d y$
h:=h1*d x+h2*d y$

% The equivalent exterior system:
factor d;
w(1) := d z(-1) + z(-1)*a + z(-2)*b - c;
w(2) := d z(-2) + z(-1)*f + z(-2)*g - h;
indexrange 1,2;
factor z;
% The integrability conditions:

integ(k) := d w(k) ^ w(1) ^ w(2);

clear a,b,c,f,g,h,x,y,w(k),integ(k),z(k);
remfdomain a1,a2,b1,c1,c2,f1,f2,g1,g2,h1,h2;

% Problem:
```

```
% --------
% Calculate the PDE's for the generators of the d-theta symmetries of
% the Lagrangian system of the planar Kepler problem.
% c.f. W.Sarlet, F.Cantrijn, Siam Review 23, 467, 1981
% Verify that time translation is a d-theta symmetry and calculate the
% corresponding integral.

pform {t,q(k),v(k),lam(k),tau,xi(k),eta(k)}=0,theta=1,f=0,
      {l,glq(k),glv(k),glt}=0;

tvector gam,y;

indexrange 1,2;

fdomain tau=tau(t,q(k),v(k)),xi=xi(t,q(k),v(k)),f=f(t,q(k),v(k));

l := 1/2*(v(1)**2 + v(2)**2) + m/r$      % The Lagrangian.

pform r=0;
fdomain r=r(q(k));
let @(r,q 1)=q(1)/r,@(r,q 2)=q(2)/r,q(1)**2+q(2)**2=r**2;

lam(k) := -m*q(k)/r;                                % The force.

gam := @ t + v(k)*@(q(k)) + lam(k)*@(v(k))$

eta(k) := gam _| d xi(k) - v(k)*gam _| d tau$

y  := tau*@ t + xi(k)*@(q(k)) + eta(k)*@(v(k))$     % Symmetry generator.

theta := l*d t + @(l,v(k))*(d q(k) - v(k)*d t)$

factor @;

s := y |_ theta - d f$

glq(k) := @(q k) _| s;
glv(k) := @(v k) _| s;
glt := @(t) _| s;

% Translation in time must generate a symmetry.
xi(k) := 0;
tau := 1;
```

```
glq k := glq k;
glv k := glv k;
glt;

% The corresponding integral is of course the energy.
integ := - y _| theta;


clear l,lam k,gam,eta k,y,theta,s,glq k,glv k,glt,t,q k,v k,tau,xi k;
remfdomain r,f,tau,xi;

% Problem:
% --------
% Calculate the "gradient" and "Laplacian" of a function and the "curl"
% and "divergence" of a one-form in elliptic coordinates.


coframe e u = sqrt(cosh(v)**2 - sin(u)**2)*d u,
        e v = sqrt(cosh(v)**2 - sin(u)**2)*d v,
        e phi = cos u*sinh v*d phi;

pform f=0;

fdomain f=f(u,v,phi);

factor e,^;
on rat,gcd;
order cosh v, sin u;
% The gradient:
d f;

factor @;
% The Laplacian:
# d # d f;

% Another way of calculating the Laplacian:
-#vardf(1/2*d f^#d f,f);

remfac @;

% Now calculate the "curl" and the "divergence" of a one-form.

pform w=1,a(k)=0;
```

```
fdomain a=a(u,v,phi);

w := a(-k)*e k;
% The curl:
x := # d w;

factor @;
% The divergence:
y := # d # w;


remfac @;
clear x,y,w,u,v,phi,e k,a k;
remfdomain a,f;


% Problem:
% --------
% Calculate in a spherical coordinate system the Navier Stokes equations.

coframe e r=d r, e theta =r*d theta, e phi = r*sin theta *d phi;
frame x;

fdomain v=v(t,r,theta,phi),p=p(r,theta,phi);

pform v(k)=0,p=0,w=1;

% We first calculate the convective derivative.

w := v(-k)*e(k)$

factor e; on rat;

cdv := @(w,t) + (v(k)*x(-k)) |_ w - 1/2*d(v(k)*v(-k));

%next we calculate the viscous terms;

visc := nu*(d#d# w - #d#d w) + mu*d#d# w;

% Finally we add the pressure term and print the components of the
% whole equation.

pform nasteq=1,nast(k)=0;
```

```
nasteq := cdv - visc + 1/rho*d p$

factor @;

nast(-k) := x(-k) _| nasteq;

remfac @,e;

clear v k,x k,nast k,cdv,visc,p,w,nasteq,e k;
remfdomain p,v;


% Problem:
% --------
% Calculate from the Lagrangian of a vibrating rod the equation of
% motion and show that the invariance under time translation leads
% to a conserved current.

pform {y,x,t,q,j}=0,lagr=2;

fdomain y=y(x,t),q=q(x),j=j(x);

factor ^;

lagr := 1/2*(rho*q*@(y,t)**2 - e*j*@(y,x,x)**2)*d x^d t;

vardf(lagr,y);

% The Lagrangian does not explicitly depend on time; therefore the
% vector field @ t generates a symmetry. The conserved current is

pform c=1;
factor d;

c := noether(lagr,y,@ t);

% The exterior derivative of this must be zero or a multiple of the
% equation of motion (weak conservation law) to be a conserved current.

remfac d;

d c;

% i.e. it is a multiple of the equation of motion.
```

```
clear lagr,c,j,y,q;
remfdomain y,q,j;

% Problem:
% --------
% Show that the metric structure given by Eguchi and Hanson induces a
% self-dual curvature.
% c.f. T. Eguchi, P.B. Gilkey, A.J. Hanson, "Gravitation, Gauge Theories
% and Differential Geometry", Physics Reports 66, 213, 1980

for all x let cos(x)**2=1-sin(x)**2;

pform f=0,g=0;
fdomain f=f(r), g=g(r);

coframe   o(r) = f*d r,
      o(theta) = (r/2)*(sin(psi)*d theta - sin(theta)*cos(psi)*d phi),
        o(phi) = (r/2)*(-cos(psi)*d theta - sin(theta)*sin(psi)*d phi),
        o(psi) = (r/2)*g*(d psi + cos(theta)*d phi);

frame e;


pform gamma(a,b)=1,curv2(a,b)=2;
index_symmetries gamma(a,b),curv2(a,b): antisymmetric;

factor o;

gamma(-a,-b) := -(1/2)*( e(-a) _| (e(-c) _| (d o(-b)))
        -e(-b) _| (e(-a) _| (d o(-c)))
        +e(-c) _| (e(-b) _| (d o(-a))) )*o(c)$


curv2(-a,b) := d gamma(-a,b) + gamma(-c,b)^gamma(-a,c)$

let f=1/g,g=sqrt(1-(a/r)**4);

pform chck(k,l)=2;
index_symmetries chck(k,l): antisymmetric;

% The following has to be zero for a self-dual curvature.

chck(k,l) := 1/2*eps(k,l,m,n)*curv2(-m,-n) + curv2(k,l);
```

```
clear gamma(a,b),curv2(a,b),f,g,chck(a,b),o(k),e(k),r,phi,psi;
remfdomain f,g;

% Example: 6-dimensional FRW model with quadratic curvature terms in
% -------
% the Lagrangian (Lanczos and Gauss-Bonnet terms).
% cf. Henriques, Nuclear Physics, B277, 621 (1986)

for all x let cos(x)**2+sin(x)**2=1;

pform {r,s}=0;
fdomain r=r(t),s=s(t);

coframe o(t) = d t,
        o(1) = r*d u/(1 + k*(u**2)/4),
        o(2) = r*u*d theta/(1 + k*(u**2)/4),
        o(3) = r*u*sin(theta)*d phi/(1 + k*(u**2)/4),
        o(4) = s*d v1,
        o(5) = s*sin(v1)*d v2
 with metric g =-o(t)*o(t)+o(1)*o(1)+o(2)*o(2)+o(3)*o(3)
               +o(4)*o(4)+o(5)*o(5);

frame e;

on nero; factor o,^;

riemannconx om;

pform curv(k,l)=2,{riemann(a,b,c,d),ricci(a,b),riccisc}=0;

index_symmetries curv(k,l): antisymmetric,
                 riemann(k,l,m,n): antisymmetric in {k,l},{m,n}
                                   symmetric in {{k,l},{m,n}},
                 ricci(k,l): symmetric;

curv(k,l) := d om(k,l) + om(k,-m)^om(m,l);

riemann(a,b,c,d) := e(d) _| (e (c) _| curv(a,b));

% The rest is done in the Ricci calculus language,

ricci(-a,-b) := riemann(c,-a,-d,-b)*g(-c,d);
```

```
riccisc := ricci(-a,-b)*g(a,b);

pform {laglanc,inv1,inv2} = 0;

index_symmetries riemc3(k,l),riemri(k,l),
                 hlang(k,l),einst(k,l): symmetric;

pform {riemc3(i,j),riemri(i,j)}=0;

riemc3(-i,-j) := riemann(-i,-k,-l,-m)*riemann(-j,k,l,m)$
inv1 := riemc3(-i,-j)*g(i,j);
riemri(-i,-j) := 2*riemann(-i,-k,-j,-l)*ricci(k,l)$
inv2 := ricci(-a,-b)*ricci(a,b);
laglanc := (1/2)*(inv1 - 4*inv2 + riccisc**2);


pform {einst(a,b),hlang(a,b)}=0;

hlang(-i,-j) := 2*(riemc3(-i,-j) - riemri(-i,-j) -
   2*ricci(-i,-k)*ricci(-j,K) +
   riccisc*ricci(-i,-j) - (1/2)*laglanc*g(-i,-j));

% The complete Einstein tensor:

einst(-i,-j) := (ricci(-i,-j) - (1/2)*riccisc*g(-i,-j))*alp1 +
hlang(-i,-j)*alp2$

alp1 := 1$
factor alp2;

einst(-i,-j) := einst(-i,-j);

clear o(k),e(k),riemc3(i,j),riemri(i,j),curv(k,l),riemann(a,b,c,d),
      ricci(a,b),riccisc,t,u,v1,v2,theta,phi,r,om(k,l),einst(a,b),
      hlang(a,b);

remfdomain r,s;

% Problem:
% --------
% Calculate for a given coframe and given torsion the Riemannian part and
% the torsion induced part of the connection. Calculate the curvature.

% For a more elaborate example see E.Schruefer, F.W. Hehl, J.D. McCrea,
```

```
% "Application of the REDUCE package EXCALC to the Poincare gauge field
% theory of gravity", GRG Journal, vol. 19, (1988)  197--218

pform {ff, gg}=0;

fdomain ff=ff(r), gg=gg(r);

coframe o(4) = d u + 2*b0*cos(theta)*d phi,
        o(1) = ff*(d u + 2*b0*cos(theta)*d phi) + d r,
        o(2) = gg*d theta,
        o(3) = gg*sin(theta)*d phi
 with metric g = -o(4)*o(1)-o(4)*o(1)+o(2)*o(2)+o(3)*o(3);

frame e;

pform {tor(a),gwt(a)}=2,gamma(a,b)=1,
      {u1,u3,u5}=0;

index_symmetries gamma(a,b): antisymmetric;

fdomain u1=u1(r),u3=u3(r),u5=u5(r);

tor(4) := 0$

tor(1) := -u5*o(4)^o(1) - 2*u3*o(2)^o(3)$

tor(2) := u1*o(4)^o(2) + u3*o(4)^o(3)$

tor(3) := u1*o(4)^o(3) - u3*o(4)^o(2)$

gwt(-a) := d o(-a) - tor(-a)$

% The following is the combined connection.
% The Riemannian part could have equally well been calculated by the
% RIEMANNCONX statement.

gamma(-a,-b) := (1/2)*( e(-b) _| (e(-c) _| gwt(-a))
               +e(-c) _| (e(-a) _| gwt(-b))
                       -e(-a) _| (e(-b) _| gwt(-c)) )*o(c);

pform curv(a,b)=2;
index_symmetries curv(a,b): antisymmetric;
factor ^;
```

```
curv(-a,b) := d gamma(-a,b) + gamma(-c,b)^gamma(-a,c);

clear o(k),e(k),curv(a,b),gamma(a,b),theta,phi,x,y,z,r,s,t,u,v,p,q,c,cs;
remfdomain u1,u3,u5,ff,gg;

showtime;
end;
```