DifferentialForms Documentation

Release 1.1.0

Kurt Pagani <nilqed@gmail.com>

Dec 22, 2016

Contents:

CHAPTER

ONE

1 THEORY

1.0 Introduction

The package `DifferentialForms` (in file `dform.spad`) builds on the domain `DeRhamComplex`. In the following section we give a brief overview of the functions that are going to be implemented. The focus is on precise definitions of the notions, since those may be varying in the literature. In section (2) we will describe the exported functions and how they work, in section (3) some short implementation notes will be given and finally the last section is devoted to some examples.

1.1 Definitions

Let $\mathcal{M}$ be a n-dimensional manifold (sufficiently smooth and orientable). To each point $P \in \mathcal{M}$ there is a neighborhood which can be diffeomorphically mapped to some region in $\mathbb{R}^n$, with coordinates

$$x_1(P'), \ldots, x_n(P')$$

for all $P' \in \mathcal{U}(P) \subset \mathcal{M}$. The tangent space $T_{P'}(\mathcal{M})$ at the point P' is a vector space, that is spanned by the basis

$$e_1(P'), \ldots, e_n(P')$$

which also is often denoted by

$$\partial_1, \ldots, \partial_n = \frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_n}.$$

A tangent vector v has the form

$$v = \sum_{j=1}^{n} v^j e_j.$$

The cotangent space $T_{P'}(\mathcal{M})^\star$ is the vector space of linear functionals

$$\alpha : T_{P'}(\mathcal{M}) \to \mathbb{R},$$

spanned by the basis $e^1(P'), \ldots, e^n(P')$ which (corresponding to the basis ∂_j) is also denoted by

$$dx^1, \ldots, dx^n.$$

The latter notation indicates the dependency on the moving point P'. The dual basis is by definition comprised of those linear functionals such that

$$e^j(e_k) = \delta^j_k.$$

Therefore we have

$$\alpha(v) = \alpha\left(\sum_{j=1}^{n} v^j e_j\right) = \sum_{j=1}^{n} v^j \alpha(e_j) = \sum_{j=1}^{n} v^j \alpha_j,$$

where $\alpha = \sum_{j=1}^{n} \alpha_j e^j$.

1.1.1 Inner product of differential forms (dot)

Let g_x be a symmetric $n \times n$ matrix which is nondegenerate (i.e. $\det(g_x) \neq 0$). The index x indicates that this matrix depends on the coordinates $x_1(P), \dots, x_n(P)$ and may be varying from point to point. If this dependency is smooth (enough) we speak of a (pseudo-) Riemannian metric (locally). This way we get an isomorphism between tangent vectors and 1-forms (= covectors):

$$\alpha_j = g_{jk} v^k, \qquad v^j = g^{jk} \alpha_j.$$

Clearly, $\sum_k g^{jk} g_{kl} = \delta_l^j$, in other words (g^{jk}) is the inverse of g. The metric g defines an *inner product* of vectors

$$g(v, w) = \langle v, w \rangle := g_{ij} v^i w^j$$

and by duality also on 1-forms:

$$g^{-1}(\alpha, \beta) = \langle \alpha, \beta \rangle := g^{ij} \alpha_i \beta_j.$$

Now, this inner product is extended to arbitrary p-forms by

$$\langle \alpha_1 \wedge \dots \wedge \alpha_p, \beta_1 \wedge \dots \wedge \beta_p \rangle := \det(\langle \alpha_i, \beta_j \rangle), \qquad (1 \leqslant i, j \leqslant p),$$

and linearity.

1.1.2. The volume form η (volumeForm)

The Riemannian *volume form* η is (by definition) given by the n-form

$$\eta = \sqrt{|\det g|} e^1 \wedge \dots \wedge e^n = \sqrt{|\det\ g|} dx^1 \wedge \dots \wedge dx^n.$$

This definition makes sense because by a (orientation preserving) change of coordinates $\sqrt{\det g}$ transforms like the component of a n-form.

1.1.3. Hodge dual (hodgeStar)

The *Hodge dual* of a differential p-form β is the (n - p)-form $\star\beta$ such that

$$\alpha \wedge \star\beta = \langle \alpha, \beta \rangle \eta$$

holds, for all p-forms α. The linear operator $(\star)$ is called the *Hodge star operator*. By the *Riesz representation theorem* the Hodge dual is uniquely defined by the expression above.

Warning

Flanders [4] defines the Hodge dual by the equality

$$\lambda \wedge \mu = \langle \star\lambda, \mu \rangle \eta$$

[4] Harley Flanders and Mathematics. Differential Forms with Applications to the Physical Sciences. Dover Pubn Inc, Auflage: Revised. edition.

where λ is a p-form and μ a (n - p)-form. This can result in different signs (actually $\star_F = s(g)\star$, where $s(g)$ is the sign of the determinant of g).

The generally adopted definition (2016) is the one given at the beginning of this subsection.

The components of $\star\beta$ are

$$(\star\beta)_{j_1,\dots,j_{n-p}} = \frac{1}{p!}\varepsilon_{i_1,\dots,i_p,j_1,\dots,j_{n-p}}\sqrt{|\det g|}g^{i_1k_1}\dots g^{i_pk_p}\beta_{k_1,\dots,k_p}$$

what is equal to

$$\frac{1}{p!\sqrt{|\det g|}}\varepsilon^{k_1,\dots,k_p,l_1,\dots,l_{n-p}}g_{j_1l_1}\dots g_{j_{n-p},l_{n-p}}\beta_{k_1,\dots,k_p}.$$

1.1.4 Interior product (interiorProduct)

The *interior product* of a vectorfield v and a p-form α is a (p -1)-form $i_v(\alpha)$ such that

$$i_v(\alpha)(v_1,\dots,v_{p-1}) = \alpha(v,v_1,\dots,v_{p-1})$$

holds, for all vectorfields $v_1,\dots,v_{p-1}$. Therefore, the components of $i_v(\alpha)$ are

$$i_v(\alpha)_{j_1,\dots,j_{p-1}} = v^j\alpha_{j,j_1,\dots,j_{p-1}}.$$

One can express the interior product by using the $\star$-operator. Let α be the 1-form defined by the equation

$$\alpha(w) = g(v,w), \forall w.$$

That means in components: $\alpha_j = g_{jk}v^k$, thus we have

$$i_v(\beta) = (-)^{p-1}\star^{-1}(\alpha\wedge\star\beta).$$

Clearly, the interior product is independent of any metric, whereas the Hodge operator is **not**! So, usually one should not use the Hodge operator to compute the interior product.

We will use the fact that the interior product is an *antiderivation*, which allows a recursive implementation.

1.1.5 The Lie derivative (lieDerivative)

The *Lie derivative* with respect to a vector field v can be calculated (and defined) using Cartan's formula:

$$\mathcal{L}_v\alpha = di_v(\alpha) + i_v(d\alpha).$$

There are other ways to define $\mathcal{L}_v\alpha$, however, it is convenient to compute it this way when d and i_v are already at hand.

1.1.6 The CoDifferential δ (codifferential)

The *codifferential* δ is defined on a p-form as follows:

$$\delta = (-1)^{n(p-1)+1}s(g)\star d\star$$

where g is the metric and $s(g)$ is related to the **signature** of $s(g)$ as described next.

1.1.7 The sign of a metric $s(g)$ (s)

The signature of a metric g is defined as the difference of the number of positive (p) and negative (q) eigenvalues, i.e:

$$\mathrm{signature(g)} = p - q$$

and the **sign** functions **s** is defined as:

$$s(g) = (-1)^{\frac{n - \mathrm{signature(g)}}{2}}$$

Since, as we always assume, g is non-degenerate, we have $p + q = n$, and consequently:

$$s(g) = (-1)^q = \operatorname{sign}\det(g)$$

1.1.8 The inverse Hodge star $\star^{-1}$ (invHodgeStar)

Applying the Hodge star operator twice on a p-form twice we get the identity up to sign:

$$\star \circ \star \omega_p = (-1)^{p(n-p)}\, s(g)\, \omega_p.$$

Therefore

$$\star^{-1} \omega_p = (-1)^{p(n-p)}\, s(g)\, \star \omega_p.$$

1.1.9 The Hodge-Laplacian Δ_g (hodgeLaplacian)

The **Hodge-Laplacian** also known as *Laplace-de Rham operator* is defined on any manifold equipped with a (pseudo-) Riemannian metric g and is given by

$$\Delta_g = d \circ \delta + \delta \circ d$$

Note that in the Euclidean case $\Delta_g = -\Delta$, where latter is the ordinary Laplacian.

Bibliography

References

CHAPTER

TWO

2 EXPORT

2.0 Package Details

Package Name DifferentialForms

Abbreviation DFORM

Source files dform.spad

Dependent on DeRhamComplex (DERHAM)

```
DifferentialForms(R,v)

R: Join(Ring,Comparable)    -- e.g. Integer
v: List Symbol              -- e.g. [x,y,z] or [x[0],x[1],x[2]]

X ==> Expression R          -- function Ring
```

For the examples following, we choose `R=Integer` and `v=[x[0],x[1],x[2],x[3]]`, and the abbreviation `M:=DFORM(R,v)`.

2.1 The metric g

Some functions expect the metric `g` as a parameter. Generally this will be provided by an invertible square matrix `g:SquareMatrix(#v,X)`.

For the examples following, we choose the Minkowski metric::

```
g := diagonalMatrix([-1,1,1,1])@SquareMatrix(4,Integer)
```

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Type: SquareMatrix(4,Integer)

2.2 Exported Functions

The function **baseForms** return the basis one forms, while **coordVector** returns a list of the coordinates. The function **coordSymbols** also returns the coordinates, however, as symbols only (convenient when used by **D**).

```
SMR ==> SquareMatrix(#v,X)
DRC ==> DeRhamComplex(R,v)

dx:=baseForms()$M      -- [dx[0],...,dx[3]]
x:=coordVector()$M     -- [x[0],...,x[3]]
xs:=coordSymbols()$M  -- as above but as List Symbol (for differentiation)
```

2.2.1 Volume Form

volumeForm Given a metric g the function returns the corresponding volume element of the Riemannian (pseudo-) manifold.

```
volumeForm : SquareMatrix(#v,X) -> DeRhamComplex(R,v)

volumeForm(g)$M
```

$$dx_0\, dx_1\, dx_2\, dx_3$$

Type: DeRhamComplex(Integer,[x[0],x[1],x[2],x[3]])

2.2.1 Scalar Product

dot Compute the inner product of two differential forms with respect to the metric g.

```
dot : (SMR,DRC,DRC) -> X

dot(g,dx.1*dx.2,dx.1*dx.2)$M   -- note dx.1 corresponds to dx[0].
```

$$-1$$

Type: Expression(Integer)

2.2.2 Hodge Star Operator

hodgeStar Compute the Hodge dual form of a differential form with respect to a metric g.

```
hodgeStar : (SMR,DRC) -> DRC

hodgeStar(g,dx.2 * dx.3)
```

$$dx_0\, dx_3$$

Type: DeRhamComplex(Integer,[x[0],x[1],x[2],x[3]])

2.2.3 Interior Product

interiorProduct Calculate the interior product $i_X(a)$ of the vector field X with the differential form a.

```
interiorProduct : (Vector(X),DRC) -> DRC

interiorProduct(vector x, dx.1*dx.3)$M
```

$$x_0\, dx_2 - x_2\, dx_0$$

Type: DeRhamComplex(Integer,[x[0],x[1],x[2],x[3]])

2.2.4 Lie Derivative

lieDerivative Calculates the Lie derivative $\mathcal{L}_X(a)$ of the differential form a with respect to the vector field X.

```
lieDerivative : (Vector(X),DRC) -> DRC

lieDerivative(vector x, dx.1 * dx.3 * dx.4)
```

$$3\, dx_0\, dx_2\, dx_3$$

Type: DeRhamComplex(Integer,[x[0],x[1],x[2],x[3]])

2.2.5 Projection

proj Project to homogeneous terms of degree p.

```
NNI ==> NonNegativeInteger
proj : (NNI,DRC) -> DRC

proj(2, 2*dx.1 + dx.2*dx.3 - dx.3*dx.4)
```

$$-dx_2\, dx_3 + dx_1\, dx_2$$

Type: DeRhamComplex(Integer,[x[0],x[1],x[2],x[3]])

2.2.6 Monomials

monomials List all monomials of degree p (p in 1..n). This is a basis for Λ_p^n.

```
monomials : NNI -> List DRC

monomials(3)$M
```

$$[dx_0\, dx_1\, dx_2,\ dx_0\, dx_1\, dx_3,\ dx_0\, dx_2\, dx_3,\ dx_1\, dx_2\, dx_3]$$

Type: List DeRhamComplex(Integer,[x[0],x[1],x[2],x[3]])

2.2.7 Atomize Basis Term

atomizeBasisTerm Given a basis term, return a list of the generators (atoms).

```
atomizeBasisTerm : DRC -> List DRC

atomizeBasisTerm(dx.1 * dx.2 * dx.4)
```

$$[dx_0,\ dx_1,\ dx_3]$$

Type: List(DeRhamComplex(Integer,[x[0],x[1],x[2],x[3]]))

2.2.8 Conjugate Basis Term

conjBasisTerm Return the complement of a basis term with respect to the Euclidean volume form.

```
conjBasisTerm : DRC -> DRC

conjBasisTerm dx.4
```

$$dx_0\, dx_1\, dx_2$$

Type: DeRhamComplex(Integer,[x[0],x[1],x[2],x[3]])

2.2.9 Scalar and Vector Field

vectorField Generate a generic vector field named by a given symbol.

covectorField Generate a generic co-vector field named by a given symbol.

scalarField Generate a generic scalar field named by a given symbol.

```
vectorField   : Symbol -> List X
covectorField : Symbol -> List DRC
scalarField   : Symbol -> X

vectorField(Q)$M
scalarField(f)$M
```

$$[Q_1\,(x_0,\; x_1,\; x_2,\; x_3),\; Q_2\,(x_0,\; x_1,\; x_2,\; x_3),\; Q_3\,(x_0,\; x_1,\; x_2,\; x_3),\; Q_4\,(x_0,\; x_1,\; x_2,\; x_3)]$$

Type: List(Expression(Integer))

$$f\,(x_0,\; x_1,\; x_2,\; x_3)$$

Type: Expression(Integer)

2.2.10 Miscellaneous Functions

A *zero form* with symbol s can be generated by

```
zeroForm : Symbol -> DRC

zeroForm(s)$M
```

$$s\,(x_0,\; x_1,\; x_2,\; x_3)$$

Type: DeRhamComplex(Integer,[x[0],x[1],x[2],x[3]])

A synonym for the **exteriorDerivative** is the common operator **d**:

```
d : DRC -> DRC

d zeroForm(f)$M
```

$$f_{,4}\,(x_0,\; x_1,\; x_2,\; x_3)\, dx_3 + f_{,3}\,(x_0,\; x_1,\; x_2,\; x_3)\, dx_2 +$$
$$f_{,2}\,(x_0,\; x_1,\; x_2,\; x_3)\, dx_1 + f_{,1}\,(x_0,\; x_1,\; x_2,\; x_3)\, dx_0$$

The special zero forms **0** and **1** can be generated by

```
one :  -> DRC
zero : -> DRC

zero()$M
one()$M
```

There are also some special multiplication operators which allow to deal with a kind *vector valued* forms (actually lists):

```
_* : (List X, List DRC)   -> DRC
_* : (List DRC, List DRC) -> DRC

Note: the lists must have dimension #v.

For instance:

x * dx
```

$$x_3\,dx_3 + x_2\,dx_2 + x_1\,dx_1 + x_0\,dx_0$$

An example for the second case:

```
dx*[hodgeStar(g,dx.j)$M for j in 1..4]
```

$$2\,dx_0\,dx_1\,dx_2\,dx_3$$

Type: DeRhamComplex(Integer,[x[0],x[1],x[2],x[3]])

CHAPTER

THREE

3 IMPLEMENTATION

3.0 Implementation Notes

In this section some implementation details will be described. This is a ongoing process and might be improved.

3.1 Internal Representation

Differential forms are represented as `List Record(k:EAB,c:R)` where each basic term is represented as `Record(k:EAB,c:R)`, for instance:

$$h(x,y,z)\ dz + g(x,y,z)\ dy + f(x,y,z)\ dx$$

maps to

```
[[k= [0,0,1],c= h(x,y,z)],[k= [0,1,0],c= g(x,y,z)],[k= [1,0,0],c= f(x,y,z)]]
```

or, another example:

$$c(x,y,z)\ dy\ dz + b(x,y,z)\ dx\ dz + a(x,y,z)\ dx\ dy$$

goes to

```
[[k= [0,1,1],c= c(x,y,z)],[k= [1,0,1],c= b(x,y,z)],[k= [1,1,0],c= a(x,y,z)]]
```

It is easily seen that for n generators $x_1,\dots,x_n$ the term $dx_{j_p}\wedge\dots\wedge dx_{j_q}$ is represented by $[k=[a_{i_1},\dots,a_{i_n}],c=\pm 1]$ where $a_{i_s}\in\{0,1\}$ depending on whether dx_{i_s} is contained in the term or not. The (local) function **terms** sends a differential form ω to the representation $r(\omega)$. Note that the operations are destructive, this means that one has to copy the objects in order to get new ones (mere assignment inherits all previous changes).

Note The interpreter normalizes the basic terms according to increasing generators, i.e for example: $dx_3\wedge dx_2$ will be stored as $-dx_2\wedge dx_3$, whereby the signum of the permutation is calculated and transferred to the `c`-field in the record.

Example:

```
terms(dx3*dx2) -> [[k= [0,1,1],c= - 1]]
```

3.2 dot :: inner product

Given a (pseudo)-Riemannian metric g, the scalar product of two basic terms of the same degree is given by

$$\langle dx_{i_1}\dots dx_{i_p}, dx_{j_1}\dots dx_{j_p}\rangle = \det(\langle dx_{i_k}, dx_{j_l}\rangle) = \det(g^{-1}(i_k,j_l)),$$

whereby $1 \leqslant k, l \leqslant p$. Note that $g^{-1}(i_k, j_l) = g^{i_k j_l}$ is the inverse of $g_{i_k j_l}$ (raised indexes as usual). In other words, the scalar product is inherited from the dual vector space of the space where the coordinates $(x_1, \dots, x_n)$ live, and is continued by linearity. By the way, terms of different **degree** are considered to be *orthogonal* to each other.

Example:

```
[x,y,z], G = matrix(G[i,j]) = g^{- 1},
```

$$\langle dx \wedge dy, dy \wedge dz\rangle = \langle dx, dy\rangle\langle dy, dz\rangle - \langle dx, dz\rangle\langle dy, dy\rangle = G[1,2]G[2,3] - G[1,3]G[2,2]$$

The corresponding EAB's are `[1,1,0]` and `[0,1,1]`. If we define a function **pos** which gives the positions of 0 or 1 respectively, the example tells us:

```
pos([1,1,0],1)=[1,2] and pos([0,1,1],1)=[2,3]
```

so that the direct product of the two resulting lists gives the desired minor:

```
[1,2]x[2,3]=[[1,2],[1,3],[2,2],[2,3]] =>
```

$$\begin{vmatrix} G_{12}G_{13} \\ G_{22}G_{23} \end{vmatrix}.$$

This essentially comprises the method we will use to compute the scalar product w.r.t symmetric matrices g and two basic terms of equal degree.

Local function: **dot2**

- compute the inverse of tmverbatim{g}.
- build the tmverbatim{pos} lists.
- build the minor and apply `determinant`.

Actually there are two functions **dot1** and **dot2**, where the former is used when the metric g is *diagonal* (which is equivalent to the basis vectors being orthogonal) because the performance might be better if the dimension of the space is huge.

3.3 hodgeStar :: Hodge dual

In this new version we have removed the first method (3.3.1) because the performance difference is not as significant as we thought in the first place, at least not for $n \leq 7$. However, we save the method in case someone has to deal with really high space dimensions.

3.3.1 Diagonal, non-degenerated g

If g is a diagonal matrix then the components of $\star\beta$ reduce to

$$(\star\beta)_{j_1,\dots,j_{n-p}} = \frac{1}{p!}\varepsilon_{k_1,\dots,k_p,j_1,\dots,j_{n-p}}\sqrt{|\det g|}[g^{k_1k_1}\dots g^{k_pk_p}]\beta_{k_1,\dots,k_p}$$

which implies that $(j_1, \dots, j_{n-p})$ must be the complement of $(k_1, \dots, k_p)$ in $\{1, 2, \dots, n\}$ and

$$\star(dx_{k_1} \wedge \dots \wedge dx_{k_p}) = C dx_{j_1} \wedge \dots \wedge dx_{j_{n-p}}$$

for some (yet unknown) factor `C` which must actually be equal to the right hand side of the component formula above. When we recollect the internal representation of $dx_{k_1} \wedge \dots \wedge dx_{k_p}$ as EAB then it is easy to get the complement by flipping the 0 and 1. Define a function **flip** such that

$$\text{flip}(dx_{k_1} \wedge \dots \wedge dx_{k_p}) = dx_{j_1} \wedge \dots \wedge dx_{j_{n-p}}$$

then by using the Hodge formula with $\alpha = \beta = dx_{k_1} \wedge \ldots \wedge dx_{k_p}$ we get using $\star\alpha = C\,\mathrm{flip}(\alpha)$:

$$C\alpha \wedge \mathrm{flip}(\alpha) = \langle\alpha, \alpha\rangle\eta = \langle\alpha, \alpha\rangle\sqrt{|\det(g)|}dx_1 \wedge \ldots \wedge dx_n.$$

Since $\alpha \wedge \mathrm{flip}(\alpha)$ is a n-form, the function **leadingCoefficient** returns the one and only coefficient. Thus we can calculate `C` to

$$C = \frac{\langle\alpha, \alpha\rangle\sqrt{|\det(g)|}}{\mathtt{leadingCoefficient}(\alpha \wedge \mathrm{flip}(\alpha))}.$$

In **SPAD** syntax this looks like:

$$\mathtt{C} = \frac{\mathtt{dot}(\alpha, \alpha) \star \mathtt{sqrt}\,(\mathtt{abs}\,(\,\mathtt{determinant}\,(\mathtt{g})))}{\mathtt{leadingCoefficient}\,(\alpha \star \mathtt{flip}(\alpha))}.$$

This way the interpreter saved us the tedious computation of the permutation signatures. Moreover, we have not to care whether the metric g is positive or negative definite.

3.3.2 General case

Let J denote an ordered multi-index and $J_\sharp$ its dual. Then a generic p-vector may be written as

$$\beta = \sum_{|J|=p} b^J\, e_J.$$

Thus by definition we obtain:

$$\alpha \wedge \star\beta = (\alpha, \beta)\,\eta \Rightarrow e_J \wedge \star\beta = (e_J, \beta)\,\eta$$

Since $\star\beta$ is a (n-p)-form, we get:

$$\star\beta = \sum_{|K|=n-p} a^K e_K \Rightarrow \sum_{|K|=n-p} a^K e_J \wedge e_K = \sum_{|I|=p} b^I(e_J, e_I) = \sum_{|I|=p} g_{JI} b^I\eta = b_J\eta.$$

Now the term $e_J \wedge e_K$ is non-zero only if $K = J_\sharp$, therefore

$$a^{J_\sharp} = \sqrt{g}\,\epsilon(J) \sum_{|I|=p} g_{JI} b^I$$

where $e_J \wedge e_{J_\sharp} = \epsilon(J)\,\eta$ defines ϵ.

If we choose $\beta = e_M$ we finally get

$$\star e_M = \sqrt{g} \sum_{|J|=p} \epsilon(J)\, g_{JM}\, e_{J_\sharp}.$$

This formula will be used to compute the Hodge dual for *monomials*. We define a function **hodgeBT**, in pseudo-code:

```
hodgeStarBT(dx[M])= sqrt(g)*
     SUM[J] {eps(dx[J])*dot(g,dx[J],dx[M])*conjBasisTerm(dx[j])}
```

which then allows to compute the Hodge dual of any form by simple recursion:

```
hodgeStar(g:SMR,x:DRC):DRC ==
  x=0$DRC => x
  leadingCoefficient(x) * hodgeStarBT(g,leadingBasisTerm(x)) + _
    hodgeStar(g, reductum(x))
```

3.4 interiorProduct :: Interior product

In this newer version we have replaced the method which uses the Hodge operator. Instead we used the fact that the interior product is an *antiderivation*, actually the unique antiderivation of degree -1 on the exterior algebra such that $i_X(\alpha) = \alpha(X)$:

$$i_X(\beta \wedge \gamma) = i_X(\beta) \wedge \gamma) + (-1)^{\mathtt{deg}\,\beta}\, \beta \wedge i_X(\gamma)$$

This also allows an easy implementation by recursion.

3.5 lieDerivative :: Lie derivative

Here we use *Cartan's formula* (see 1.1.5), so that there is not much to say.

```
lieDerivative(w:Vector X,x:DRC):DRC ==
  a := exteriorDifferential(interiorProduct(w,x))
  b := interiorProduct(w, exteriorDifferential(x))
  a+b
```

3.6 proj :: Projection

Since the elements of `DeRhamComplex` are in

$$X = \bigoplus_{p=0}^{n} \Omega^p(V)$$

it is convenient to have a function `proj` : $\{0,\ldots,n\} \times X \to X$ which returns the projection on the homogeneous component $\Omega^p(V)$. The implementation is straightforward when using the internals of EAB. Probably there are better ways to do this, especially by using exported functions only.

```
** deprecated **
proj(x,p) ==
  t:List REA := x::List REA
  idx := [j for j in 1..#t | #pos(t.j.k,1)=p]
  s := [copy(t.j) for j in idx::List(NNI)]
  convert(s)$DRC
```

NEW

In the new version we actually replaced the function above by the following recursive one:

```
proj(p,x) ==
  x=0 => x
  homogeneous? x and degree(x)=p => x
  a:=leadingBasisTerm(x)
  if degree(a)=p then
    leadingCoefficient(x)*a + proj(p, reductum x)
  else
    proj(p, reductum x)
```

NOTE We have changed the order of arguments from (DRC,NNI) to (NNI,DRC) because this corresponds more to the usual nomenclature of projections.

CHAPTER

FOUR

4 USAGE

4.0 Examples

In this chapter some examples are provided.

4.1 Calculus in $\mathbb{R}^3$

We will prove the following identities (see the summary, 4.1.4, for details):

$$\begin{aligned} d\,f &= [\texttt{grad}\,f]_1 \\ d\,[T]_1 &= [\texttt{curl}\,T]_2 \\ d\,[T]_2 &= [\texttt{div}\,T]_3 \end{aligned}$$

Let M denote our differential graded algebra on $\mathbb{R}^3$. In FriCAS we can express this as

```
M ==> DFORM(INT,[x,y,z])
```

$$\texttt{DifferentialForms}(\texttt{Integer}, [\texttt{x}, \texttt{y}, \texttt{z}])$$

Type: Type

The list of available methods can be seen by

```
)show M

DifferentialForms(Integer,[x,y,z]) is a package constructor.
Abbreviation for DifferentialForms is DFORM
This constructor is exposed in this frame.
------------------------------- Operations --------------------------------

coordSymbols : () -> List(Symbol)
...
baseForms : () -> List(DeRhamComplex(Integer,[x,y,z]))
coordVector : () -> List(Expression(Integer))
.
.
.
```

The position vector $P = (x, y, z)$ and the basis of one forms can be obtained by

```
P:=coordVector()$M
```

$$[x, y, z]$$

Type: List(Expression(Integer))

and

```
dP:=baseForms()$M
```

$$[dx, dy, dz]$$

Type: List(DeRhamComplex(Integer,[x,y,z]))

This way we can call the coordinates as $P.i$ and the basis one forms as $dP.i$. Of course, we can also use dx, dy, dz directly when we set

```
[dx,dy,dz]:=baseForms()$M
```

or when we use the generators of the domain `DeRhamComplex` itself:

```
dx:=generator(1)$DERHAM(INT,[x,y,z])
dy:= ...
```

The first method, however, is quite convenient when using indexed coordinates and also because we can form expressions like

```
P * dP
```

$$z\ dz + y\ dy + x\ dx$$

Type: DeRhamComplex(Integer,[x,y,z])·

4.1.1 Gradient

There are many ways to create a zero form, one of those is

```
f := zeroForm(f)$M
```

$$f(x, y, z)$$

Type: DeRhamComplex(Integer,[x,y,z])

Now we apply the exterior differential operator to f:

```
d f
```

$$f_{,3}\left(x,\ y,\ z\right)\ dz + f_{,2}\left(x,\ y,\ z\right)\ dy + f_{,1}\left(x,\ y,\ z\right)\ dx$$

Type: DeRhamComplex(Integer,[x,y,z])

The coefficients of df are just

```
[coefficient(d f, dP.j) for j in 1..3]
```

$$[f_{,1}\left(x,\ y,\ z\right),\ f_{,2}\left(x,\ y,\ z\right),\ f_{,3}\left(x,\ y,\ z\right)]$$

Type: List(Expression(Integer))

the components of the gradient vector ∇f of f.

4.1.2 Curl

Let T be a generic vector field on $M = \mathbb{R}^3$:

```
T := vectorField(T)$M
```

$$[T_1(x,\ y,\ z),\ T_2(x,\ y,\ z),\ T_3(x,\ y,\ z)]$$

Type: List(Expression(Integer))

Then we build the general one form τ:

```
tau := T * dP
```

Now we apply the exterior differential operator d:

```
d tau
```

$$\left(T_{3,2}(x,y,z)-T_{2,3}(x,y,z)\right)\,dy\,dz+\left(T_{3,1}(x,y,z)-T_{1,3}(x,y,z)\right)\,dx\,dz+\left(T_{2,1}(x,\ y,\ z)-T_{1,2}(x,\ y,\ z)\right)\,dx\,dy$$

Type: DeRhamComplex(Integer,[x,y,z])

Next, we want to extract the coefficients:

```
[coefficient(d tau, m) for m in monomials(2)$M]
```

$$\left[T_{2,1}(x,y,z)-T_{1,2}(x,y,z),\ T_{3,1}(x,y,z)-T_{1,3}(x,y,z),\ T_{3,2}(x,y,z)-T_{2,3}(x,y,z)\right]$$

The (well known) **curl** is defined as

$$\texttt{curl}(T) = \nabla\times T = \left(\frac{\partial T_3}{\partial y}-\frac{\partial T_2}{\partial z},\frac{\partial T_1}{\partial z}-\frac{\partial T_3}{\partial x},\frac{\partial T_2}{\partial x}-\frac{\partial T_1}{\partial y}\right)$$

```
curl(V) == [D(V.3,y)-D(V.2,z),D(V.1,z)-D(V.3,x),D(V.2,x)-D(V.1,y)]
```

We now **claim** that the following identity holds:

$$d(T\,dP) = \star(\texttt{curl}(V)\,dP)$$

where $\star$ denotes the Hodge star operator with respect to the Euclidean metric

```
g:=diagonalMatrix([1,1,1])
```

$$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}$$

To prove it we just have to test:

```
test( d(T*dP) = hodgeStar(g,curl(T)*dP)$M )
```

```
true
```

Type: Boolean

4.1.3 Divergence

Again, let T be a generic vector field on $M = \mathbb{R}^3$, then the divergence is defined by

$$\texttt{div}(T) = \nabla \bullet T = \tfrac{\partial T_1}{\partial x} + \tfrac{\partial T_2}{\partial y} + \tfrac{\partial T_3}{\partial z}.$$

When we calculate

```
d hodgeStar(g, T*dP)$M
```

we get the 3-form

$$\left(T_{3,3}\left(x,\ y,\ z\right) + T_{2,2}\left(x,\ y,\ z\right) + T_{1,1}\left(x,\ y,\ z\right)\right)\,dx\,dy\,dz$$

4.1.4 Summary

Let us summarize what we have obtained above. We use the following notation for the mapping of scalar functions and vector fields to differential forms:

$$f \to [f]_0$$
$$T \to [T]_1$$

where the index denotes the degree of the form. Moreover, we define another pair of forms by applying the Hodge operator:

$$[T]_2 = \star[T]_1$$
$$[f]_3 = \star[f]_0$$

So we can state the general identities:

$$d\,f = [\nabla\,f]_1$$
$$d\,[T]_1 = [\texttt{curl}\,T]_2$$
$$d\,[T]_2 = [\texttt{div}\,T]_3$$

4.1.5 Hodge duals

To conclude this example, we are going to calculate a table for the Hodge duals of the monomials.

```
g:=diagonalMatrix([1,1,1])::SquareMatrix(3,INT)

[[hodgeStar(g,m)$M for m in monomials(j)$M] for j in 0..3]
```

$$[[dx\,dy\,dz],\ [dy\,dz,\ -dx\,dz,\ dx\,dy],\ [dz,\ -dy,\ dx],\ [1]]$$

Type: List(List(DeRhamComplex(Integer,[x,y,z])))

Thus we get the following table:

α	$\star\alpha$	$\star\star\,\alpha$
1	$dx \wedge dy \wedge dz$	1
dx	$dy \wedge dz$	dx
dy	$-dx \wedge dz$	dy
dz	$dx \wedge dy$	dz

By the way, this method can be applied in any dimension for any metric.

4.2 Faraday 2-form

The free electromagnetic field can be described by a 2-form **F** in Minkowski space. This form - also known as Faraday 2-form - is given by

$$F=B_1\ dy\wedge dz+B_2\ dz\wedge dx+B_3\ dx\wedge dy+E_1\ dx\wedge dt+E_2\ dy\wedge dt+E_3\ dz\wedge dt$$

where we here use the **cgs** system and **E**, **B** denote the classical fields (see the example in the documentation of `DeRhamComplex`).

To represent **F** in FriCAS we have to choose space-time variables x, y, z, t, in the correct order, and g will be the Minkowski metric:

```
v := [x,y,z,t]

g := diagonalMatrix([-1,-1,-1,1])::SquareMatrix(4,INT)

M := DFORM(INT,v)

R ==> EXPR(INT)
```

Instead of x, y, z, t we also could have chosen x_0, x_1, x_2, x_3 for instance. Now we need the coordinates and basis one forms:

Important The order of the variables must coincide with that in the metric g. That means for example, for t, x, y, z the positive `1` comes first.

```
X := coordVector()$M

dX := baseForms()$M
```

We also need the field **E** and **B**, but this time we will not choose the `vectorField` function because we only need three components:

```
E := [operator E[i] for i in 1..3]
B := [operator B[i] for i in 1..3]
```

Eventually we can build **F**:

```
F := (B.1 X)*dX.2*dX.3 + (B.2 X)*dX.3*dX.1 + (B.3 X)*dX.1*dX.2 +_
     (E.1 X)*dX.1*dX.4 + (E.2 X)*dX.2*dX.4 + (E.3 X)*dX.3*dX.4
```

$$E_3(x,\ y,\ z,\ t)\ dz\ dt+E_2(x,\ y,\ z,\ t)\ dy\ dt+B_1(x,\ y,\ z,\ t)\ dy\ dz+$$
$$E_1\left(x,\ y,\ z,\ t\right)dx\ dt-B_2\left(x,\ y,\ z,\ t\right)dx\ dz+B_3\left(x,\ y,\ z,\ t\right)dx\ dy$$

Type: DeRhamComplex(Integer,[x,y,z,t])

We apply the exterior differential operator **d** to **F**:

```
d F
```

$$\left(E_{3,2}(x,\ y,\ z,\ t)-E_{2,3}(x,\ y,\ z,\ t)+B_{1,4}(x,\ y,\ z,\ t)\right)\ dy\ dz\ dt+$$
$$\left(E_{3,1}\left(x,\ y,\ z,\ t\right)-E_{1,3}\left(x,\ y,\ z,\ t\right)-B_{2,4}\left(x,\ y,\ z,\ t\right)\right)dx\ dz\ dt+$$
$$\left(E_{2,1}\left(x,\ y,\ z,\ t\right)-E_{1,2}\left(x,\ y,\ z,\ t\right)+B_{3,4}\left(x,\ y,\ z,\ t\right)\right)dx\ dy\ dt+$$
$$\left(B_{3,3}\left(x,\ y,\ z,\ t\right)+B_{2,2}\left(x,\ y,\ z,\ t\right)+B_{1,1}\left(x,\ y,\ z,\ t\right)\right)dx\ dy\ dz$$

Type: DeRhamComplex(Integer,[x,y,z,t])

We see at once that the first three terms of the sum correspond to the vector

$$\nabla \times \mathbf{E} + \frac{\partial \mathbf{B}}{\partial t}$$

and the fourth term is

$$\nabla \bullet \mathbf{B}.$$

Actually, all terms are zero by two of the *Maxwell* equations. Consequently we have shown (the well known fact)

$$d\mathbf{F} = 0$$

Now let us apply the $\star$-operator to **F**, which is also a 2-form:

```
%F := hodgeStar(g,F)$M
```

$$B_3(x,\ y,\ z,\ t)\ dz\ dt + B_2(x,\ y,\ z,\ t)\ dy\ dt - E_1(x,\ y,\ z,\ t)\ dy\ dz + \\ B_1\left(x,\ y,\ z,\ t\right)\ dx\ dt + E_2\left(x,\ y,\ z,\ t\right)\ dx\ dz - E_3\left(x,\ y,\ z,\ t\right)\ dx\ dy$$

Type: DeRhamComplex(Integer,[x,y,z,t])

Now, as before:

```
d %F
```

$$\left(-E_{1,4}(x,\ y,\ z,\ t) + B_{3,2}(x,\ y,\ z,\ t) - B_{2,3}(x,\ y,\ z,\ t)\right)\ dy\ dz\ dt + \\ \left(E_{2,4}\left(x,\ y,\ z,\ t\right) + B_{3,1}\left(x,\ y,\ z,\ t\right) - B_{1,3}\left(x,\ y,\ z,\ t\right)\right)\ dx\ dz\ dt + \\ \left(-E_{3,4}\left(x,\ y,\ z,\ t\right) + B_{2,1}\left(x,\ y,\ z,\ t\right) - B_{1,2}\left(x,\ y,\ z,\ t\right)\right)\ dx\ dy\ dt + \\ \left(-E_{3,3}\left(x,\ y,\ z,\ t\right) - E_{2,2}\left(x,\ y,\ z,\ t\right) - E_{1,1}\left(x,\ y,\ z,\ t\right)\right)\ dx\ dy\ dz$$

Type: DeRhamComplex(Integer,[x,y,z,t])

Again, we see that the first three terms correspond to

$$-\frac{\partial \mathbf{E}}{\partial t} + \nabla \times \mathbf{B}$$

while the last one corresponds to:

$$-\nabla \bullet \mathbf{E}$$

Thus, in vacuum, these are the second pair of *Maxwell's* equation and we have:

$$d \star \mathbf{F} = 0$$

To conclude this example we will compute the quantities (4-forms):

$$\mathbf{F} \wedge \mathbf{F} \ \text{ and } \ \mathbf{F} \wedge \star\mathbf{F}.$$

Recalling the definition of the Hodge dual it is sufficient (in principle) to compute the scalar product $\langle F, F\rangle$:

```
dot(g,F,F)$M
```

$$-E_3(x,\ y,\ z,\ t)^2 - E_2(x,\ y,\ z,\ t)^2 - E_1(x,\ y,\ z,\ t)^2 + \\ B_3\left(x,\ y,\ z,\ t\right)^2 + B_2\left(x,\ y,\ z,\ t\right)^2 + B_1\left(x,\ y,\ z,\ t\right)^2$$

Type: Expression(Integer)

and $\langle F, \star F\rangle$:

```
dot(g,F,%F)$M
```

$$-2\ B_3(x,y,z,t)\ E_3(x,y,z,t) - 2\ B_2(x,y,z,t)\ E_2(x,y,z,t) - 2\ B_1(x,y,z,t)\ E_1(x,y,z,t)$$

Type: Expression(Integer)

Indeed, we can *test* the defining identity, e.g. for the first case:

```
test(F * %F = dot(g,F,F)$M * volumeForm(g)$M)
```

```
true
```

Type: Boolean

4.3 Some Examples from *Maple*

Examples from Maple.

4.3.1 5-dimensional Manifold

First create a 5-dimensional manifold M and define a metric tensor g on the tangent space of M:

```
v:=[x[j] for j in 1..5]
M:=DFORM(INT,v)
g:=diagonalMatrix([1,1,1,1,1])::SquareMatrix(5,INT)
dX:=baseForms()$M
```

```
hodgeStar(g,dX.1)$M
```

$$dx_2\ dx_3\ dx_4\ dx_5$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4],x[5]])

```
hodgeStar(g,dX.2)$M
```

$$-dx_1\ dx_3\ dx_4\ dx_5$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4],x[5]])

```
hodgeStar(g,dX.2*dX.3)$M
```

$$dx_1\ dx_4\ dx_5$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4],x[5]])

```
hodgeStar(g,dX.2*dX.4)$M
```

$$-dx_1\ dx_3\ dx_5$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4],x[5]])

```
hodgeStar(g,dX.2*dX.3*dX.4)$M
```

$$-dx_1\ dx_5$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4],x[5]])

We see an exact match with the published results.

4.3.2 General metric (2-dim)

To show the dependence of the Hodge star operator upon the metric, we consider a general metric g on a 2-dimensional manifold.

```
v:=[x,y]
M:=DFORM(INT,v)
R ==> EXPR INT
g:=matrix([[a::R,b],[b,c]])::SquareMatrix(2,R)
[dx,dy]:=baseForms()$M
```

```
hodgeStar(g,dx)$M
```

$$\frac{c\,\sqrt{abs\left(a\,c-b^2\right)}}{a\,c-b^2}\,dy+\frac{b\,\sqrt{abs\left(a\,c-b^2\right)}}{a\,c-b^2}\,dx$$

Type: DeRhamComplex(Integer,[x,y])

```
hodgeStar(g,dy)$M
```

$$-\frac{b\,\sqrt{abs\left(a\,c-b^2\right)}}{a\,c-b^2}\,dy-\frac{a\,\sqrt{abs\left(a\,c-b^2\right)}}{a\,c-b^2}\,dx$$

Type: DeRhamComplex(Integer,[x,y])

```
f := hodgeStar(g,dx*dy)$M
```

$$\frac{\sqrt{abs\left(a\,c-b^2\right)}}{a\,c-b^2}$$

Type: DeRhamComplex(Integer,[x,y])

```
hodgeStar(g,f)$M
```

$$\frac{abs\left(a\,c-b^2\right)}{a\,c-b^2}\,dx\,dy$$

Type: DeRhamComplex(Integer,[x,y])

4.3.3 Laplacian

The Laplacian of a function with respect to a metric g can be calculated using the exterior derivative and the Hodge star operator. Generally, the following identity holds:

$$\Delta = d\circ\delta+\delta\circ d$$

where $\delta := (-1)^p \star^{-1} d\star$ is the **codifferential** to be applied on a p-form (resulting in a (p-1)-form). Therefore, the Laplacian applied to a function f (zero form) is:

$$\Delta f = \delta\circ df = \star^{-1} d\star df = \star d\star df.$$

```
v:=[r,u]  -- polar coordinates
M:=DFORM(INT,v)
R ==> EXPR INT
g:=matrix([[1,0],[0,r^2]])::SquareMatrix(2,R)
[dr,du]:=baseForms()$M
```

A function on M can easiliy be defined by

```
f:=zeroForm(f)$M
```

$$f\left(r,\ u\right)$$

Type: DeRhamComplex(Integer,[r,u])

We translate the formula:

```
hodgeStar(g, d hodgeStar(g,d f)$M)$M
```

$$\frac{abs\left(r^2\right)\ f_{,2,2}\left(r,\ u\right)+r^2\ abs\left(r^2\right)\ f_{,1,1}\left(r,\ u\right)+r\ abs\left(r^2\right)\ f_{,1}\left(r,\ u\right)}{r^4}$$

Type: DeRhamComplex(Integer,[r,u])

Simplifying yields for M:

$$\Delta_M f=\frac{\partial^2 f}{\partial r^2}+\frac{1}{r}\frac{\partial f}{\partial r}+\frac{1}{r^2}\frac{\partial^2 f}{\partial u^2}$$

4.3.4 Lie derivative

```
v:=[x[i] for i in 1..3]
M:=DFORM(INT,v)
dX:=baseForms()$M
V:=vectorField(V)$M
f:=scalarField(f)$M
```

```
lieDerivative(V,dX.1)
```

$$V_{1,3}(x_1,\,x_2,\,x_3)\ dx_3+V_{1,2}(x_1,\,x_2,\,x_3)\ dx_2+V_{1,1}(x_1,\,x_2,\,x_3)\ dx_1$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3]])

```
lieDerivative(V,f*dX.1)
```

$$\mathcal{L}_{\mathcal{V}}\,\{\,\lceil\S_\infty$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3]])

```
lieDerivative(V,f*dX.1*dX.2)
```

$$\mathcal{L}_{\mathcal{V}}\,\{\,\lceil\S_\infty\wedge\lceil\S_\in$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3]])

4.4 More examples (way of working)

Setup

```
)clear all

  All user variables and function definitions have been cleared.
```

```
n:=4 -- dim of base space (n>=2 !)
R:=Integer  -- Ring

v:=[subscript(x,[j::OutputForm]) for j in 1..n] -- (x_1,..,x_n)

M:=DFORM(R,v)

-- basis 1-forms and coordinate vector
dx:=baseForms()$M     -- [dx[1],...,dx[n]]
x:=coordVector()$M    -- [x[1],...,x[n]]
xs:=coordSymbols()$M  -- as above but as List Symbol (for differentiate, D)

-- operator, vector field, scalar field, symbol
a:=operator 'a         -- operator
b:=vectorField(b)$M    -- generic vector field [b1(x1..xn),...,bn(x1..xn)]
c:=vectorField(c)$M
P:=scalarField(P)$M -- scalar field P(x1,..,xn)

-- metric
g:=diagonalMatrix([1 for i in 1..n])$SquareMatrix(n,EXPR R)  -- Euclidean
h:=diagonalMatrix(c)$SquareMatrix(n,EXPR R)

-- vector field (R)
vf:=vector b
```

Macros

```
-- macros
dV(g) ==> volumeForm(g)$M
i(X,w) ==> interiorProduct(X,w)$M
L(X,w) ==> lieDerivative(X,w)$M
** w ==> hodgeStar(g,w)$M  -- don't use * instead of ** !
```

```
w:=x.1*dx.2-x.2*dx.1
```

$$x_1\ dx_2 - x_2\ dx_1$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
d w
```

$$2\ dx_1\ dx_2$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
w*w
```

$$0$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
i(vf,w)
```

$$x_1\ b_2\left(x_1,\ x_2,\ x_3,\ x_4\right) - x_2\ b_1\left(x_1,\ x_2,\ x_3,\ x_4\right)$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
L(vf,w)
```

$$\begin{gathered}\left(x_1\, b_{2,4}\left(x_1,\, x_2,\, x_3,\, x_4\right) - x_2\, b_{1,4}\left(x_1,\, x_2,\, x_3,\, x_4\right)\right) dx_4+\\ \left(x_1\, b_{2,3}\left(x_1,\, x_2,\, x_3,\, x_4\right) - x_2\, b_{1,3}\left(x_1,\, x_2,\, x_3,\, x_4\right)\right) dx_3+\\ \left(x_1\, b_{2,2}\left(x_1,\, x_2,\, x_3,\, x_4\right) - x_2\, b_{1,2}\left(x_1,\, x_2,\, x_3,\, x_4\right) + b_1\left(x_1,\, x_2,\, x_3,\, x_4\right)\right) dx_2+\\ \left(x_1\, b_{2,1}\left(x_1,\, x_2,\, x_3,\, x_4\right) - x_2\, b_{1,1}\left(x_1,\, x_2,\, x_3,\, x_4\right) - b_2\left(x_1,\, x_2,\, x_3,\, x_4\right)\right) dx_1\end{gathered}$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
d i(vf,w) + i(vf,d w)
```

$$\begin{gathered}\left(x_1\, b_{2,4}\left(x_1,\, x_2,\, x_3,\, x_4\right) - x_2\, b_{1,4}\left(x_1,\, x_2,\, x_3,\, x_4\right)\right) dx_4+\\ \left(x_1\, b_{2,3}\left(x_1,\, x_2,\, x_3,\, x_4\right) - x_2\, b_{1,3}\left(x_1,\, x_2,\, x_3,\, x_4\right)\right) dx_3+\\ \left(x_1\, b_{2,2}\left(x_1,\, x_2,\, x_3,\, x_4\right) - x_2\, b_{1,2}\left(x_1,\, x_2,\, x_3,\, x_4\right) + b_1\left(x_1,\, x_2,\, x_3,\, x_4\right)\right) dx_2+\\ \left(x_1\, b_{2,1}\left(x_1,\, x_2,\, x_3,\, x_4\right) - x_2\, b_{1,1}\left(x_1,\, x_2,\, x_3,\, x_4\right) - b_2\left(x_1,\, x_2,\, x_3,\, x_4\right)\right) dx_1\end{gathered}$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
% - L(vf,w)
```

$$0$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
dot(g,w,w)$M
```

$${x_2}^2 + {x_1}^2$$

Type: Expression(Integer)

```
d i(vf,dV(g)) -- div(b) dV
```

$$\left(b_{4,4}\left(x_1,\, x_2,\, x_3,\, x_4\right) + b_{3,3}\left(x_1,\, x_2,\, x_3,\, x_4\right) + b_{2,2}\left(x_1,\, x_2,\, x_3,\, x_4\right) + b_{1,1}\left(x_1,\, x_2,\, x_3,\, x_4\right)\right) dx_1\, dx_2\, dx_3\, dx_4$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
d (P*one()$M) -- One()?
```

$$\begin{gathered}P_{,4}\left(x_1,\, x_2,\, x_3,\, x_4\right) dx_4 + P_{,3}\left(x_1,\, x_2,\, x_3,\, x_4\right) dx_3+\\ P_{,2}\left(x_1,\, x_2,\, x_3,\, x_4\right) dx_2 + P_{,1}\left(x_1,\, x_2,\, x_3,\, x_4\right) dx_1\end{gathered}$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
i(vf,%)
```

$$\begin{gathered}b_1\left(x_1,\, x_2,\, x_3,\, x_4\right) P_{,1}\left(x_1,\, x_2,\, x_3,\, x_4\right)+\\ b_2\left(x_1,\, x_2,\, x_3,\, x_4\right) P_{,2}\left(x_1,\, x_2,\, x_3,\, x_4\right)+\\ b_3\left(x_1,\, x_2,\, x_3,\, x_4\right) P_{,3}\left(x_1,\, x_2,\, x_3,\, x_4\right)+\\ b_4\left(x_1,\, x_2,\, x_3,\, x_4\right) P_{,4}\left(x_1,\, x_2,\, x_3,\, x_4\right)\end{gathered}$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
1/dot(g,w,w)$M*w
```

$$\frac{x_1}{{x_2}^2 + {x_1}^2}\, dx_2 - \frac{x_2}{{x_2}^2 + {x_1}^2}\, dx_1$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
d %
```

$$0$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
s:=zeroForm('s)$M
```

$$s\left(x_1,\ x_2,\ x_3,\ x_4\right)$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
d s
```

$$s_{,4}\left(x_1,\ x_2,\ x_3,\ x_4\right)\,dx_4 + s_{,3}\left(x_1,\ x_2,\ x_3,\ x_4\right)\,dx_3 + \\ s_{,2}\left(x_1,\ x_2,\ x_3,\ x_4\right)\,dx_2 + s_{,1}\left(x_1,\ x_2,\ x_3,\ x_4\right)\,dx_1$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
d (** s)
```

$$0$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
** ( d s)
```

$$s_{,1}\left(x_1,\ x_2,\ x_3,\ x_4\right)\,dx_2\,dx_3\,dx_4 - s_{,2}\left(x_1,\ x_2,\ x_3,\ x_4\right)\,dx_1\,dx_3\,dx_4 + \\ s_{,3}\left(x_1,\ x_2,\ x_3,\ x_4\right)\,dx_1\,dx_2\,dx_4 - s_{,4}\left(x_1,\ x_2,\ x_3,\ x_4\right)\,dx_1\,dx_2\,dx_3$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
d (** ( d s)) -- Laplacian(s) dV
```

$$\left(s_{,1,1}\left(x_1,\ x_2,\ x_3,\ x_4\right) + s_{,2,2}\left(x_1,\ x_2,\ x_3,\ x_4\right) + s_{,3,3}\left(x_1,\ x_2,\ x_3,\ x_4\right) + s_{,4,4}\left(x_1,\ x_2,\ x_3,\ x_4\right)\right)\,dx_1\,dx_2\,dx_3\,dx_4$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
r:=sin(x.1*x.2)*one()$M
```

$$\sin\left(x_1\ x_2\right)$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
d r
```

$$x_1\ \cos\left(x_1\ x_2\right)\,dx_2 + x_2\ \cos\left(x_1\ x_2\right)\,dx_1$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
d (** ( d r))
```

$$\left(-{x_2}^2 - {x_1}^2\right)\ \sin\left(x_1\ x_2\right)\,dx_1\,dx_2\,dx_3\,dx_4$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
** (d (** ( d r)))
```

$$\left(-{x_2}^2-{x_1}^2\right)\sin\left(x_1\ x_2\right)$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
** (d (** ( d r)))::EXPR INT
```

$$\left(-{x_2}^2-{x_1}^2\right)\sin\left(x_1\ x_2\right)$$

Type: Expression(Integer)

```
eval(%,xs.1=%pi)
```

$$\left(-\pi^2-{x_2}^2\right)\sin\left(x_2\ \pi\right)$$

Type: Expression(Integer)

```
eval(%,xs.2=%pi/3)
```

$$-\frac{10\ \pi^2\sin\left(\frac{\pi^2}{3}\right)}{9}$$

Type: Expression(Integer)

```
a(P)*one()$M
```

$$a\left(P\left(x_1,\ x_2,\ x_3,\ x_4\right)\right)$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

```
d (a(P)*one()$M) -- chain diff
```

$$\begin{array}{r}P_{,4}\left(x_1,\ x_2,\ x_3,\ x_4\right)a'\left(P\left(x_1,\ x_2,\ x_3,\ x_4\right)\right)dx_4+\\ P_{,3}\left(x_1,\ x_2,\ x_3,\ x_4\right)a'\left(P\left(x_1,\ x_2,\ x_3,\ x_4\right)\right)dx_3+\\ P_{,2}\left(x_1,\ x_2,\ x_3,\ x_4\right)a'\left(P\left(x_1,\ x_2,\ x_3,\ x_4\right)\right)dx_2+\\ P_{,1}\left(x_1,\ x_2,\ x_3,\ x_4\right)a'\left(P\left(x_1,\ x_2,\ x_3,\ x_4\right)\right)dx_1\end{array}$$

Type: DeRhamComplex(Integer,[x[1],x[2],x[3],x[4]])

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search