DifferentialGeometry1

`Domain: DG1`

Kurt Pagani
`nilqed@gmail.com`

December 20, 2016

Abstract

This manual describes the FriCAS package **DifferentialGeometry1**. This package combines differential forms and cell mappings to provide methods which compute **pull backs**, **integrals** as well as some other quantities of differential forms living on a surface complex.

Contents

1 Introduction

Initially, it was planned to build this package by reusing `JET` (included in FriCAS), a sophisticated series of domains and packages written by Joachim Schue and Werner M. Seiler ([10], [11],[12],[15]). Actually, most of the functionality of this package has already been implemented in a new package `JetBundleGeometry` when we recognized that jet bundles in FriCAS are much more powerful than just serving as a helper package for local coordinate patches. Continuing the process would have us led to unfortunate dependencies on cell mappings and other domains, so to make it difficult to use higher order jet bundles in connection with differential forms. Consequently we extracted everything useful into this package. Because parts of the manual also had been written the reader will certainly recognize this here and there, although we tried to remove most unrelated topics. The package `JetBundleGeometry` will be based on a new domain, `JetDifferentialForm`[1] and will only depend on `JET`. The rest of this section may be skipped, however, it is assumed the reader is acquainted with the manuals of `DifferentialForm` and `SurfaceComplex`.

Recall that a bundle is a triple (M, X, π), where M, X are manifolds and $\pi : M \to X$ is the projection, i.e. a continuous surjective mapping from the total space (M) to the base space (X). A bundle is called trivial if M is homeomorphic to $X \times U$, where U denotes another manifold, called the fibre. A locally trivial bundle is called a fibre bundle, which is what we here are concerned with. We will use the following nomenclature:

$$x = (x_1, \ldots, x_p), \quad u = (u^1, \ldots, u^n)$$

where x are the (local) coordinates of the base manifold X and u the the ones on the fibre U. Locally the projection takes the form

$$\pi : \begin{cases} X \times U \\ (x, u) \mapsto x. \end{cases}$$

A section of the fibre bundle then has the form

$$\Phi_f : \begin{cases} X \times U \\ x \mapsto (x, f(x)). \end{cases}$$

such that $\pi \circ \Phi_f = \mathcal{I}d$. A jet bundle now may be considered as an iteration of that construction, that is we will speak of a n-th order jet bundle $M^{(n)}$ if

$$M^{(n)} = X \times (U \times U_1 \times \ldots \times U_n)$$

where the U_j are the jet (Euclidean) spaces. We will just write M for $M^{(0)}$.

[1] work in progress

1.1 Motivation

Let us recall the following domains/packages and their functionality:

1 DeRhamComplex provides differential forms as a graded ring in a given set of variables:

$$\texttt{DeRhamComplex}(\texttt{R}, [\texttt{x}_1, \ldots, \texttt{x}_\texttt{n}]) = \bigoplus_{p=0}^{n} \Lambda^p([x_1, \ldots, x_n])$$

where $x_j \in \texttt{Expression}(\texttt{R})$, R a ring, meaning that the coefficients are from `Expression R`.

2 DifferentialForms is a package extending `DeRhamComplex` by *Riemannian metrics* g and the connected standard operations like *Hodge star* $\star_g$, δ_g, Δ_g, i_X, $\mathcal{L}_X$ and $\langle \cdot, \cdot \rangle_g$.

3 CellMap and SurfaceComplex are domains which loosely spoken will provide parametric surface patches and their formal sums (free group with integer coefficients):

$$\texttt{SurfaceComplex}(\texttt{R}, \texttt{n}) = \mathbb{Z} \cdot \bigoplus_{p=0}^{n} \Sigma^p([\$_1, \ldots, \$_n])$$

where Σ_p is the collection of $p-$cells, that is mappings from a $p-$cell into $n-$space.

Consequently we can define the boundary operator ∂ such that $T(d\varphi) = \partial T(\varphi)$ holds, where T is an element of SurfaceComplex and φ one of DeRhamComplex.

But note that the code of these domains is unrelated, that is both can be used independently. So we are in need of a package or domain which brings them together in order to utilize their mutual duality.

A second point is the need of computing *pull backs* of differential forms and - closely related - of integration of forms over *affine chains*. To achieve this we have to incorporate two (usually) different spaces of differential forms.

Eventually, the following quotations from [11] together with the remarks above induced the usage of `JetBundle` instead of creating a new domain:

> ... The implementation of both is somewhat rudimentary, as we hope that some day `AXIOM` will contain a reasonable environment for differential geometric calculations and then it should

> be used instead of some special domains like the two mentioned. ... The implementation of the differential geometric domains `VectorField` (abbreviated by `VF`) and `Differential (DIFF)` are fairly primitive. They should be considered as a temporary hack, until AXIOM contains a better environment for differential geometric calculations. ...

We may interpret `JetBundle` in the first place as a local coordinate patch (chart) with independent variables x and dependent ones u. Then we will be able to pull forms in $u-$ space back to x-space. In other words we will only use a zero order jet bundle for the moment. In a second step the interplay might be extended to higher order bundles such that the deficiencies mentioned in the above quotes may be mitigated.

2 Theory

For the general theory about pulling back - and integration of differential forms we refer to one of the excellent books, e.g. ([6],[2],[3],[4]), cited in the bibliography. We recall two important facts about pulling back differential forms α, β by smooth maps f:

$$\begin{aligned} f^\star(\alpha \wedge \beta) &= f^\star\alpha \wedge f^\star\beta, \\ d\, f^\star\alpha &= f^\star(d\,\alpha). \end{aligned}$$

The implementation of `pullBack` is based on this. Furthermore, we have for $f : Q \to M$ the identity

$$\int_M \omega = \int_Q f^\star\omega,$$

on which `integrate` is based on. Usually Q is a $p-$cell and M a $p-$surface (or surface complex). In other words, f is `S.map` and Q is `S.dom` when S is of type `CellMap`. Since we have the restriction `#(S.dom)`$\leq n$, where n is the dimension of the range of `S.map`, the mapping f may also be described by a set of equations (see further below). This essentially is all we need to compute most of the usual differential quantities that occur in connection with local and global surface theories. As remarked in the introduction, this package will only serve as a simple link between the domain of affine chains `SurfaceComplex` and the domain of co-chains `DeRhamComplex`. More sophisticated methods will be implemented in the forthcoming package `JetBundleGeometry`.

2.1 Local Coordinates

To set up the package we have to provide two lists of symbols which will denote the dependent- and the independent coordinates, for example

```
M ==> DG1([x,y],[u,v,w])
```

means that u, v, w are *dependent* coordinates, that is functions of the *independent* coordinates x, y. Note that M is just a package selector without any information how this dependencies are defined. However, M will provide two spaces of differential forms, namely

```
DeRhamComplex(Integer,[x,y]) -- e.g. a(x,y) dx dy
DeRhamComplex(Integer,[u,v,w]) -- e.g. b(u,v,w) du dv.
```

When we - for instance - think of a parametric surface

$$(x, y) \mapsto (u(x, y), v(x, y), w(x, y))$$

we immediately see how the pull back of the $2-$form $b(u, v, w)\, du \wedge dv$ to the parameter domain looks like:

$$b(u(x, y), v(x, y), w(x, y))du(x, y) \wedge dv(x, y),$$

where $du(x, y) \wedge dv(x, y)$ expands to

$$(u_x dx + u_y dy) \wedge (v_x dx + v_y dy) = (u_x v_y - v_x u_y)dx \wedge dy$$

of course.

2.2 $p-$Surfaces

Recall that a $p-$surface is a mapping from a compact cell Q of dimension p into $n-$space, where we always assume $p \leq n$. Staying with the parametric surface picture above, we have $p = 2$, and $n = 3$, and we may speak of a $2-$surface. Since $p = n - 1$ it actually is a hyper-surface but this is in no way a must. Usually n is chosen as big as necessary in order to obtain an *embedding* of the (generalized) surface. Remembering *Nash's* embedding theorem, we can (almost) always isometrically embed a Riemannian manifold into Euclidean space if n is sufficiently big.

Back to the example, if we want to integrate our form over the $2-$surface, we will specify a cell mapping of type `CellMap(Integer,3)` as follows:

```
CM3 ==> CellMap(Integer,3)
Q:=[a..b,c..d] -- 2-cell
S:=cellMap(Q,q+->[F(q.1,q.2),G(q.1,q.2),H(q.1,q.2)])$CM3
```

We see that S is formulated independent of the coordinates in M, only if we apply S to (x, y), we will get the connection between the variables u, v, w and x, y. Now we have

$$\int_{S(Q)} b(u,v,w)\, du \wedge dv = \int_Q S^\star(b(u,v,w)\, du \wedge dv)$$

or, using the results above,

$$= \int_Q b(u(x,y), v(x,y), w(x,y))(u_x v_y - v_x u_y) dx \wedge dy$$

what reduces to an ordinary double integral over the cell Q:

$$\int_c^d \int_a^b b(u(x,y), v(x,y), w(x,y))(u_x v_y - v_x u_y) dx dy.$$

This is just how these integrals over cell maps are computed:

```
integrate(b(u,v,w)*du*dv,S)$M
```

2.3 SurfaceComplex

Most if not all operations extend to type `SurfaceComplex` by linearity. It is, however, the responsibility of the user to ensure that the results make sense. One has to bear in mind that a differential forms in FriCAS need not be homogeneous and the same holds for the free abelian group of cell mappings. As a warning example recall the *Möbius band* in $\mathbb{R}^3$ which is probably the simplest instance of a non-orientable surface.

3 Exported Functions

We will provide an example to each function. For this reason let us reuse M form the last section and define a new constructor N:

```
M ==> DG1([x,y],[u,v,w])
N ==> DG1([x[0],x[1],x[2]],[y[1],y[2],y[3],y[4])
```

Furthermore, we will denote by XS and YS the lists of symbols given to `DG1`, e.g. `XS=[x,y]` in case of M, and by X and Y the corresponding lists of variables.

3.1 Independent Variables: `indVars()`

To get the list of independent variables of `DG1(...)` call

```
indVars()$DG1(XS,US) => X
```

```
(1) -> indVars()$N

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

(2) -> indVars()$M

   (2)  [x,y]
Type: List(Expression(Integer))
```

3.2 Dependent Variables: `depVars()`

To get the list of dependent variables of `DG1(...)` call

```
depVars()$DG1(XS,US) => U
```

```
(1) -> depVars()$N

   (1)  [y ,y ,y ,y ]
          1  2  3  4
Type: List(Expression(Integer))

(2) -> depVars()$M

   (2)  [u,v,w]
Type: List(Expression(Integer))
```

3.3 In-/Dependent Symbols: `indSymbols, depSymbols`

The symbol lists can be recovered are frequently necessary when differentiating, i.e. when using the operator **D**.

```
indSymbols()$DG1(XS,US) => XS
depSymbols()$DG1(XS,US) => US
```

```
(1) -> depSymbols()$M

   (1)  [u,v,w]
Type: List(Symbol)

(2) -> indSymbols()$N

   (2)  [x ,x ,x ]
          0  1  2
Type: List(Symbol)
```

3.4 In-/Dependent 1−Forms: `indForms, depForms`

To obtain the lists of dependent or independent one forms1− forms:

```
indForms()$DG1(XS,US) => dX
depForms()$DG1(XS,US) => dU
```

```
(1) -> depForms()$M

   (1)  [du,dv,dw]
Type: List(DeRhamComplex(Integer,[u,v,w]))

(2) -> depForms()$N

   (2)  [dy ,dy ,dy ,dy ]
          1   2   3   4
Type: List(DeRhamComplex(Integer,[y[1],y[2],y[3],y[4]]))
```

3.5 Types involved: `getTypes()`

This function returns a list of all form and surface types created in the package. This might be useful/convenient to define corresponding macros.

```
getTypes()$DG1(XS,US) => [DERHAM(INT,XS),DERHAM(INT,US),
                          DFORM(INT,XS), DFORM(INT,US)]
```

```
(1) -> t:=getTypes()$M

 LISP output:
(UNPRINTABLE UNPRINTABLE UNPRINTABLE UNPRINTABLE)
Type: List(Type)

-- #t = 4

(2) -> t.1

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

(2) -> t.4

   (2)  DifferentialForms(Integer,[u,v,w])
Type: Type
```

3.6 Get Types by Name: `getDxxx`

Each of the types of `getType` can also be retrieved as follows:

```
    getDRCX : () -> Type
      -- returns the asscoiated DeRhamComplex in the
      -- independent variables indVars(),
```

```
    -- i.e DeRhamComplex(Integer,XS).

  getDRCU : () -> Type
    -- returns the asscoiated DeRhamComplex in the
    -- dependent variables depVars(),
    -- i.e DeRhamComplex(Integer,US).

  getDFX : () -> Type
    -- returns the asscoiated DifferentialForms package in
    -- the independent variables indVars(),
    -- i.e DifferentialForms(Integer,XS).

  getDFU : () -> Type
    -- returns the asscoiated DifferentialForms package in
    -- the dependent variables depVars(),
    -- i.e DifferentialForms(Integer,US).
```

```
(1) -> DFU ==> getDFU()$N
Type: Void

(2) -> DFU

   (2)  DifferentialForms(Integer,[y[1],y[2],y[3],y[4]])
Type: Type
```

3.7 Pull back of a form: pullBack

The function pullBack is overloaded as follows:

```
  pullBack : (CellMap(Integer,#us),DRCU) -> DRCX
    -- pullBack(S,w) computes the pull back of the differential
    -- form w by the cell map S.

  pullBack : (SurfaceComplex(Integer,#us),DRCU) -> DRCX
    -- pullBack(S,w) computes the pull back of the differential
    -- form w by all cellMaps in a surface complex.

  pullBack : (List Equation R,DRCU) -> DRCX
    -- pullBack(S,w) computes the pull back of the differential
    -- form w by the section given as a list of equations of
    -- the form U=f(X).
```

3.8 Integration of a form: integrate

```
integrate : (DRCU, CellMap(Integer,#us)) -> R
 -- integrate(w,S) computes the integral of a differential form
 -- w over a cell map S.

integrate : (DRCU, SurfaceComplex(Integer,#us)) -> R
 -- integrate(w,SC) computes the integral of a differential form
 -- w over a surface complex SC.
```

4 Appendix

References

[1] Singer, I. M., and Thorpe, J. A.*Lecture Notes on Elementary Topology and Geometry*, Scott, Foresman and Company.

[2] Spivak, M. *Calculus on Manifolds*, W. A. Benjamin, Inc., New York, 1965.

[3] Ralph Abraham, Jerrold E.Marsden and Tudor Ratiu.Manifolds, *Tensor Analysis, and Applications*. Springer, Auflage: 2nd Corrected ed. 1988. Corr. 2nd printing 1993 edition.

[4] Henri Cartan. *Differential Forms*. Dover Pubn Inc.

[5] Herbert Federer. *Geometric Measure Theory*. Springer, Reprint of the 1st ed. Berlin, Heidelberg, New York 1969 edition.

[6] Harley Flanders, *Differential Forms with Applications to the Physical Sciences*. Dover Pubn Inc, Revised. edition.

[7] L. A. Lambe and D. E. Radford. *Introduction to the Quantum Yang-Baxter Equation and Quantum Groups:An Algebraic Approach*. Springer, 1997 edition.

[8] Walter Rudin and RudinWalter.*Principles of Mathematical Analysis*. McGraw Hill Book Co, Revised. edition.

[9] Hassler Whitney. *Geometric Integration Theory*, Princeton Mathematical Series, No. 21. Literary Licensing, LLC.

[10] J. Schü, W.M. Seiler, and J. Calmet. Algorithmic methods for Lie pseudogroups. In N. Ibragimov, M. Torrisi, and A. Valenti, editors, *Proc. Modern Group Analysis: Advanced Analytical and Computational Methods in Mathematical Physics*, pages 337–344, Acireale (Italy), 1992. Kluwer, Dordrecht 1993.

[11] W.M. Seiler. Applying AXIOM to partial differential equations. Internal Report 95-17, Universität Karlsruhe, Fakultät für Informatik, 1995. Available from: `http://citeseerx.ist.psu.edu`

[12] W.M. Seiler. *Analysis and Application of the Formal Theory of Partial Differential Equations*. PhD thesis, School of Physics and Materials, Lancaster University, 1994.

[13] W.M. Seiler. Involution and symmetry reductions. Preprint 1995.

[14] W.M. Seiler and R.W. Tucker. Involution and constrained dynamics I: The Dirac approach. *J. Phys. A*, to appear, 1995.

[15] W.M. Seiler and Jacques Calmet *JET - An AXIOM Environment for Geometric Computations with Differential Equations* Electronic Proc. IMACS Conference on Applications of Computer Algebra, Albuquerque 1995, M. Wester, S. Steinberg, M. Jahn (eds.)