Updates to Differential Geometry in Maple 16

With over 250 commands, the DifferentialGeometry package allows sophisticated computations from basic jet calculus to the realm of the mathematics behind general relativity. In addition, 19 differential geometry lessons, from beginner to advanced level, and 6 tutorials illustrate the use of the package in applications.

In Maple 16, the DifferentialGeometry package introduces important new functionalities for working with abstractly defined differential forms, general relativity, and Lie algebras.

- You can now work with differential forms which are defined without reference to any underlying system of coordinates.
- The new MetricSearch assistant provides a simple method for searching the database of solutions to the Einstein equations.
- New commands for analyzing the geometric properties of spacetime are introduced.
- A new tutorial for differential geometry and general relativity has been added.
- Twenty-four new commands allow working with simple and semi-simple Lie algebras, including performing a complete analysis of the structure theory for any semi-simple Lie algebra.

Jump to examples

Details

Abstractly defined differential forms (See example)

- The functionality of the command DGsetup has been extended to create a new computational environment for working with abstract differential forms, that is, differential forms which are defined without reference to any underlying system of coordinates. Structure equations for the abstract differential forms can be included in the new calling sequences for DGsetup. The capabilities of other commands in the DifferentialGeometry package, notably DGinfo, ExteriorDerivative, Hook and LieDerivative, have been extended to work with abstract differential forms.   These capabilities extend those available in the difforms package.

The Maplet MetricSearch

- The Library sub-package contains a new command MetricSearch which provides a simple method for searching the DifferentialGeometry database of solutions to the Einstein equations.

General Relativity (See example)

- The Tensor sub-package contains 4 new commands for analyzing the geometric properties of space-times.
- The commands RainichConditions and RainichElectromagneticField can be used to determine if a given space-time is an electro-vac space-time and to find the electromagnetic field for that space-time.
- Line congruences play a distinguished role in the study of space-times and the properties of such congruences can now be easily calculated with the command CongruenceProperties.
- The signature of a metric can now be computed with QuadraticFormSignature. This procedure uses a simple algorithm which does not require eigenvalue computation and works well with the Maple assuming feature.

Structure Theory of Semi-simple Lie Algebras (See example)

- Twenty-four new commands have been added to the LieAlgebra package for working with simple (and semi-simple) Lie algebras. The *DifferentialGeometry* software now contains the commands for the complete analysis of the structure theory for any semi-simple Lie algebra (defined perhaps as a matrix algebra, a Lie algebra of symmetries of a differential equation, the isometries of a metric, the automorphisms of a Lie algebra, etc).
- The command SimpleLieAlgebraData can be used define the structure equations for any of the classical real simple Lie algebras $sl(n), su(p, q)$, $su^*(n), so(p, q)$, $so^*(n), sp(n, \mathbb{R})$, $sp(p, q)$, $sp(n)$. There are three companion procedures to *SimpleLieAlgebraData.* The procedure StandardRepresentation gives a basis for the standard representation of these Lie algebras as matrices. The procedure

SimpleLieAlgebraProperties returns a table of properties for any one of these simple Lie algebras - these properties include Cartan subalgebra, root space decomposition, simple roots, positive roots, Cartan decomposition, etc. The command MatrixSubalgebra is used to create subalgebras of these classic matrix algebras.

- The command CartanSubalgebra finds a Cartan subalgebra for any given Lie algebra.
- The command RootSpaceDecomposition uses the Cartan subalgebra to produce a table which gives the root space decomposition of the Lie algebra. The roots can be extracted from the table with LieAlgebraRoots. The compact roots (purely imaginary) are found with CompactRoots. The root space decomposition is performed assuming that the Lie algebra is defined over the complex numbers. A real root space decomposition is obtained from RestrictedRootSpaceDecomposition.
- The command PositiveRoots will determine a set of positive roots from the root space decomposition. The command SimpleRoots finds the basis of simple roots from the positive roots.
- The Cartan matrix is found from the simple roots and the root space decomposition with CartanMatrix. The command CartanMatrixToStandardForm produces a re-ordering of the simple roots which transforms the Cartan matrix to standard form and identifies the complex type of the Lie algebra as "A", "B", "C", "D", "E", or "F". The command SatakeAssociate can be used to determine the real type of a given simple Lie algebra.
- The commands CartanDecomposition and CartanInvolution are available for semi-simple Lie algebras with a given matrix representation.
- The root space decomposition can be used to create gradings of a semi-simple Lie algebra or to find parabolic subalgebras.
- The signature of the Killing form can be computed with the KillingForm and QuadraticFormSignature commands.
- The Dynkin and Satake diagrams for a simple Lie algebra can be plotted with the commands DynkinDiagram and SatakeDiagram.

Query command

- The LieAlgebra Query command now supports keyword arguments "CartanDecomposition", "CartanInvolution", "CartanSubalgebra", "MatrixAlgebra", "NilRepresentation", "ParabolicSubalgebra", "RegularElement", "RootSpaceDecomposition", "SolvableRepresentation".

Examples

Example 1.

In this example we shall initialize the Lie algebra $\mathfrak{sl}(3)$ (the algebra of 3×3 trace-free matrices). For this algebra we then calculate [i] a Cartan subalgebra, [ii] the root space decomposition, [iii] the positive and simple roots, and [iv] the Cartan matrix.

```
> with(DifferentialGeometry): with(LieAlgebras): with(Tensor):
with(Tools):
```

```
> LD := SimpleLieAlgebraData("sl(3)", sl3);
```

$$LD := [[e1, e3] = e3, [e1, e4] = 2\,e4, [e1, e5] = -e5, [e1, e6] = e6, [e1, e7] = -2\,e7, [e1, e8] = -e8, [e2, e3] = -e3, [e2, e4] = e4, [e2, e5] = e5, [e2, e6] = 2\,e6, [e2, e7] = -e7, [e2, e8] = -2\,e8, [e3, e5] = e1 - e2, [e3, e6] = e4, [e3, e7] = -e8, [e4, e5] = -e6, [e4, e7] = e1, [e4, e8] = e3, [e5, e8] = -e7, [e6, e7] = e5, [e6, e8] = e2]$$

Initialize this algebra.

```
> DGsetup(LD);
```

Lie algebra: sl3

Calculate a Cartan subalgebra.

```
sl3 > CSA := CartanSubalgebra(sl3);
```

$$CSA := [e1, e2]$$

Find the root space decomposition for this algebra.

```
sl3 > RSD := RootSpaceDecomposition(CSA);
```

$$RSD := table([[2, 1] = e4, [1, -1] = e3, [-1, 1] = e5, [-1, -2] = e8, [1, 2] = e6, [-2, -1] = e7])$$

Find the positive roots and the simple roots.

```
sl3 > Rts:= LieAlgebraRoots(RSD);
```

$$Rts := \left[\begin{bmatrix}2\\1\end{bmatrix}, \begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}-1\\1\end{bmatrix}, \begin{bmatrix}-1\\-2\end{bmatrix}, \begin{bmatrix}1\\2\end{bmatrix}, \begin{bmatrix}-2\\-1\end{bmatrix}\right]$$

```
sl3 > PosRts := PositiveRoots(Rts, [<1,0>, <0, 1>]);
```

$$PosRts := \left[\begin{bmatrix}2\\1\end{bmatrix}, \begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}1\\2\end{bmatrix}\right]$$

```
sl3 > Delta := SimpleRoots(PosRts);
```

$$\Delta := \left[\begin{bmatrix}1\\-1\end{bmatrix}, \begin{bmatrix}1\\2\end{bmatrix}\right]$$

Find the Cartan matrix.

```
sl3 > CartanMatrix(Delta, RSD);
```

$$\begin{bmatrix}2 & -1\\-1 & 2\end{bmatrix}$$

Example 2.

A space-time is called an electro-vac spacetime if there exists an electromagnetic field which solves the Einstein-Maxwell field equations. The problem of deciding if a spacetime is electro-vac can be solved using the command RainichConditions and RainichElectromagneticField.

```
M > DGsetup([t, x, y, z], M);
```

$$\textit{frame name: M}$$

```
M > g := evalDG(4/3*t^2* dx &t dx + t*(exp(-2*x)* dy &t dy +
exp(2*x)*dz &t dz) - dt &t dt);
```

$$g := -dt\,dt + \frac{4}{3}\,t^2\,dx\,dx + t\,e^{-2\,x}\,dy\,dy + t\,e^{2\,x}\,dz\,dz$$

Test to see if the Rainich conditions for this spacetime hold.

```
M > RainichConditions(g);
```

$$\textit{true}$$

We conclude the spacetime is an electro-vac spacetime. Here is the electro-magnetic field.

```
M > F := RainichElectromagneticField(g);
```

$$F := \frac{2\cos(\_C1)\left(\operatorname{csgn}\left(\frac{1}{t^2}\right)+1\right)dt \wedge dx}{\sqrt{6\operatorname{csgn}\left(\frac{1}{t^2}\right)+6}} + \frac{\sin(\_C1)\sqrt{3}\operatorname{csgn}\left(\frac{1}{t^2}\right)\left(\operatorname{csgn}\left(\frac{1}{t^2}\right)+1\right)dy \wedge dz}{\sqrt{6\operatorname{csgn}\left(\frac{1}{t^2}\right)+6}}$$

```
M > F := simplify(F) assuming t > 0;
```

$$F := \frac{2}{3}\cos(\_C1)\sqrt{3}\,dt \wedge dx + \sin(\_C1)\,dy \wedge dz$$

We check that the Einstein equations are satisfied.

```
M > T := EnergyMomentumTensor("Electromagnetic", g, F);
```

$$T := \frac{1}{2}\frac{D\_tD\_t}{t^2} - \frac{3}{8}\frac{D\_xD\_x}{t^4} + \frac{1}{2}\frac{e^{2x}D\_yD\_y}{t^3} + \frac{1}{2}\frac{e^{-2x}D\_zD\_z}{t^3}$$

```
M > E := EinsteinTensor(g);
```

$$E := \frac{1}{2}\frac{D\_tD\_t}{t^2} - \frac{3}{8}\frac{D\_xD\_x}{t^4} + \frac{1}{2}\frac{e^{2x}D\_yD\_y}{t^3} + \frac{1}{2}\frac{e^{-2x}D\_zD\_z}{t^3}$$

```
M > E &minus T;
```

$$0\,D\_tD\_t$$

We check that the Maxwell equations   are satisfied.

```
M > MatterFieldEquations("Electromagnetic", g, F);
```

$$0\,D\_t,\, 0\,dt \wedge dx \wedge dy$$

Example 3.

The DifferentialGeometry package now supports extended functionality for working with abstract differential forms

Create an abstract manifold M with a function f, 1-forms α, β and a 2-form σ .

```
> DGsetup('[f = dgform(0) , alpha = dgform(1), beta = dgform(1), sigma
= dgform(2)]', [], M);
```

frame name: M

The command DGinfo gives the names of all scalars and forms which are defined.

```
M > DGinfo("AbstractForms");
```

$$[f, \alpha, \beta, \sigma]$$

Scalar products, wedge products and sums of abstract forms can be defined.

```
M > omega := evalDG(2*alpha &wedge beta + 4*sigma);
```

$$\omega := 2\,\alpha \wedge \beta + 4\,\sigma$$

The command DGinfo can also be used to extract information about the form ω.

```
M > DGinfo(omega, "ObjectFrame");
```

$$M$$

```
M > DGinfo(omega, "FormDegree");
```

$$2$$

```
M > DGinfo(omega, "CoefficientSet");
```

$$\{2, 4\}$$

```
M > DGinfo(omega , "CoefficientList", [sigma]);
```

$$[4]$$

New forms can be defined on M.

```
M > DGsetup(M, [delta = dgform(3)],[]);
```

$$\textit{updated frame: M}$$

```
M > DGinfo("AbstractForms");
```

$$[f, \alpha, \beta, \sigma, \delta]$$

We can use the DGzip and GetComponents commands with abstract forms.

```
M > Omega := evalDG([alpha &w beta, sigma]);
```

$$\Omega := [\alpha \wedge \beta, \sigma]$$

```
M > zeta := DGzip([3, 5], Omega, "plus");
```

$$\zeta := 3\,\alpha \wedge \beta + 5\,\sigma$$

```
M > GetComponents(zeta, Omega);
```

$$[3, 5]$$

We can take the exterior derivative of a form.

```
M > rho:= ExteriorDerivative(alpha);
```

$$\rho := d\,\alpha$$

The 2-form $d\alpha$ has been added to list of defined forms and is now available for subsequent computations.

```
M > DGinfo("AbstractForms");
```

$$[f, \alpha, \beta, \sigma, \delta, d\,\alpha]$$

```
M > ExteriorDerivative(rho);
```

$$0\,\alpha \wedge \sigma$$

Exterior derivatives of defined forms can be specified.

```
M > DGsetup(M, [], [d(f) = f*alpha, d(beta) = 4*sigma + 5*alpha &wedge
beta]);
```

updated frame: M

```
M > ExteriorDerivative(f*beta);
```

$$6 f \alpha \wedge \beta + 4 f \sigma$$