ISSN 0361-7688, Programming and Computer Software, 2013, Vol. 39, No. 3, pp. 135–142. © Pleiades Publishing, Ltd., 2013.
Original Russian Text © A.V. Korol'kova, D.S. Kulyabov, L.A. Sevast'yanov, 2013, published in Programmirovanie, 2013, Vol. 39, No. 3.

Tensor Computations in Computer Algebra Systems

A. V. Korol'kova, D. S. Kulyabov, and L. A. Sevast'yanov

Peoples' Friendship University of Russia, ul. Miklukho-Maklaya 6, Moscow, 117198 Russia

e-mail: sevleonid@yandex.ru

Received July 25, 2012

Abstract—This paper considers three types of tensor computations. On their basis, we attempt to formulate criteria that must be satisfied by a computer algebra system dealing with tensors. We briefly overview the current state of tensor computations in different computer algebra systems. The tensor computations are illustrated with appropriate examples implemented in specific systems: Cadabra and Maxima.

DOI: 10.1134/S0361768813030031

1. INTRODUCTION

Tensor calculations are used in many fields of physics. It should be noted that the formalism of tensor analysis manifests itself in all its might not in all areas; rather frequently, its simplified versions are used.

Each tensor operation itself is sufficiently simple. However, even standard computations have to involve many elementary operations. These operations require great care and thoroughness. That is why it is important in this field to use different simplified notations and optimized operations (for example, Penrose tensor diagrams).

One of the tasks of computer algebra systems is to free the researcher from routine operations, which is important in the case of tensor calculus.

2. MAIN APPLICATION FIELDS AND TYPES OF TENSOR NOTATION

To define the key operations with tensors, consider the main field of their application.

2.1. Nonindex Computations for Theoretical Constructs

Nonindex computations are commonly used in theoretical constructs and are often opposed to component computations. Let us see how to implement basic tensor operations in the nonindex case.

(*) *Addition of tensors.* The addition of two tensors of valence $\begin{bmatrix} p \\ q \end{bmatrix}$ yields a tensor of valence $\begin{bmatrix} p \\ q \end{bmatrix}$:

$$A + B = C. \tag{1}$$

The addition of tensors determines the structure of an Abelian group.

(*) *Multiplication of tensors.* The multiplication of tensor A with valence $\begin{bmatrix} p \\ q \end{bmatrix}$ by tensor D with valence $\begin{bmatrix} r \\ s \end{bmatrix}$ yields tensor E with valence $\begin{bmatrix} p+r \\ q+s \end{bmatrix}$:

$$A \otimes D = E. \tag{2}$$

The tensor multiplication defines the structure of a noncommutative semigroup.

(*) *Operation of Contraction.* Let us denote the operation of contraction of tensors with respect to the last indices by $\mathfrak{S}$. Then, under the action of this operation, tensor F with valence $\begin{bmatrix} p+1 \\ q+1 \end{bmatrix}$ goes into tensor G with valence $\begin{bmatrix} p \\ q \end{bmatrix}$:

$$\mathfrak{S} F = G. \tag{3}$$

(*) *Operation of permutation of indices.* This operation is necessary for specifying the symmetry of tensors (for example, a tensor commutator or anticommutator), for expanding the operation of contraction on the contraction with respect to arbitrary indices. However, the nonindex approach gives no way of identifying this operation. Yet, the simplest symmetries can be explicitly specified in the object description (in this case, one has to impose restrictions on valence to ensure uniqueness).

2.2. Vector Computations

Vector calculus is the simplest case of tensor calculus (vectors are tensors of valence one). An N-dimen-

sional vector a^n is represented as a set of components $n = \overline{1, N}$, depending on the basis, and a linear law of the transformation of components for a changing basis. The frequently used operations are the construction of various differential operators and change of the basis. The most common operators are gradient, divergence, and curl (specific to the three-dimensional space $\mathbb{R}^3$) [1, 2].

Component calculations require a basis, a metric, and connectivity (and, accordingly, a covariant derivative) to be defined. In vector calculus, it is common to use a holonomic basis, which is constructed as a set of partial derivatives of the coordinates in a tangent bundle and the dual basis as a 1-form in the cotangent bundle:

$$\vec{\delta}_i = \frac{\partial}{\partial x^i}, \quad \vec{\delta}^i = dx^i. \tag{4}$$

The connectivity and metric are constructed in such a way that the covariant derivative of the metric are equal to zero:

$$\nabla_k g_{ij} = 0. \tag{5}$$

In this case, the connectivity and metric are consistent [3].

It should be noted that vector computations often use a special nonholonomic basis that makes it possible not to distinguish between contravariant and covariant vectors and keep dimension unchanged under a coordinate transformation[1] (for details, see [1]):

$$\vec{\delta}_i = \frac{\partial}{\partial s^i}, \quad \vec{\delta}^i = ds^i, \quad ds^i = h^i_j dx^j. \tag{6}$$

Here, ds^i is the element of length with respect to a given coordinate and h^i_j are the nonholonomy coefficients (the Lamé coefficients in the case of orthogonal coordinates).

2.3. Dirac 4-Spinors

A special case of tensor objects are spinors (also called spin-tensors). Particularly, spinors are representations of the Lorentz group with a half-integer highest weight. Conventional tensors are representations with an integer highest weight.

For historical reasons, the investigations most commonly use Dirac 4-spinors, which are applied to write the Dirac equations describing fermions with spin 1/2. Dirac 4-spinors are essentially irreducible spinors for the case $n = 4$ and $s = \pm 2$, where n is the dimension of the vector space, $s = n - 2u$ is its signature, and u is the number of negative values of the diagonal metric tensor g_{ab}.

Usually, the handling with Dirac spinors is based on γ-matrices derived from the Clifford–Dirac equation [4]:

$$\gamma_{(a}\gamma_{b)} = g_{ab}\hat{I}, \tag{7}$$

where γ_a are $N \times N$ matrices, g_{ab} is a metric tensor, $\hat{I}$ is the identity $N \times N$ matrix, and N is the dimension of the spinor space:

$$N = \begin{cases} 2^{n/2}, & \text{even } n, \\ 2^{n/2-1/2}, & \text{odd } n. \end{cases} \tag{8}$$

γ-matrices are Clifford algebra elements generating a linear transformation of the spin space.

Since the γ-matrix can be regarded as the coefficients of transition from the spin space to the vector space, one should introduce the spin coefficients more strictly and write Eq. (7) as

$$\gamma^\sigma_{a\rho}\gamma^\tau_{b\sigma} + \gamma^\sigma_{b\rho}\gamma^\tau_{a\sigma} = 2g_{ab}\delta^\sigma_\rho. \tag{9}$$

The construction of a complete Clifford algebra requires also products of γ-matrices, but, in view of (7), it is suffice to consider only antisymmetrized products

$$\gamma_{ab\ldots d} := \gamma_{[a}\gamma_b \cdots \gamma_{d]}. \tag{10}$$

Also, we introduce an element γ_5:

$$\gamma_5 := \frac{i}{4!} e^{abcd}\gamma_a\gamma_b\gamma_c\gamma_d, \tag{11}$$

where e^{abcd} is an alternating tensor.

Manipulations on the γ-matrices reduce to a set of relations that follow from algebraic symmetries, such as, for example,

$$\gamma^a\gamma_a = 4\hat{I}, \tag{12}$$

$$\gamma^a\gamma^b\gamma^c\gamma_a = 4g^{bc}\hat{I}, \tag{13}$$

$$\gamma_a\gamma_b = \gamma_{ab} + g_{ab}\hat{I}, \tag{14}$$

$$\gamma_a\gamma_b\gamma_c = \gamma_{abc} + g_{ab}\gamma_c + g_{bc}\gamma_a - g_{ac}\gamma_b. \tag{15}$$

2.4. Tensor Calculations in the Theory of General Relativity

General relativity became the first physical theory requiring the entire power of differential geometry and tensor calculations [5]. These calculations involve bulky tensor constructs that can be simplified in view of the symmetry of tensors. Usually, one differentiates between monoterm and multiterm symmetries. A key element of the theory is the Riemann tensor, which has both very simple monoterm and complex multiterm symmetries like the Bianchi identities.

Monoterm symmetries correspond to simple permutation symmetries and are given by a group of permutations. Specifically, for the Riemann tensor, we have

$$R_{bacd} = -R_{abcd}, \quad R_{cdab} = R_{abcd}. \tag{16}$$

Multiterm symmetries are given by an algebra of permutations. The Bianchi identity has the form[2]:

$$R_{a(bcd)} = R_{abcd} + R_{acdb} + R_{adbc} = 0. \tag{17}$$

[1] Under this transformation, length goes into length, angle goes into angle, etc.

[2] The round brackets in (17) denote symmetrization.

The differential (second) Bianchi identity has the form[3]:

$$R_{ab(cd;\,e)} = \nabla_e R_{abcd} + \nabla_c R_{abdr} + \nabla_d R_{abec} = 0. \quad (18)$$

It is most reasonable to specify symmetries by means of the Young diagrams [6]. Here, the presence of predefined classes of tensors does not eliminate the need for an explicit specification of symmetry. For example, the Riemann tensor R_{abcd} in different sources has the symmetries $\begin{array}{|c|c|}\hline a & c \\ \hline b & d \\ \hline \end{array}$ and $\begin{array}{|c|c|}\hline a & b \\ \hline c & d \\ \hline \end{array}$.

2.5. Types of Tensor Notation

Thus, based on the above types of tensor calculations, one can specify three types of tensor notation: component notation, notation with abstract indices, and nonindex notation. Each type has its own specificity and application field.

Component indices actually turn a tensor into a set of scalar values used in specific calculations. Usually, it makes sense to operate with component indices only after simplifying the tensor expression and taking into account all of its symmetries.

The nonindex notation is often used when the researcher is interested in the symmetry of tensors rather than in the final result. However, this form of notation lacks expressiveness: the tensor is regarded as an integral entity; accordingly, only the symmetries that are related to the tensor as a whole can be considered. To operate with objects of complex structures, one has to invent new notation or add verbal explanations. It is this problem that should be treated by abstract indices [7].

Abstract indices should be regarded as an improvement of the nonindex notation of tensors. An abstract index denotes merely the fact that a tensor belongs to a certain space rather than obeys the tensor transformation rule (unlike component indices). In this case, one can consider both symmetries covering the full tensor (all its indices) and symmetries of individual groups of indices.

3. TENSOR COMPUTATIONS AND COMPUTER ALGEBRA SYSTEMS

Modern computer algebra systems are capable of solve problems of a sufficiently wide class and from different areas of knowledge. The systems can be highly specialized or with a claim to universality (a survey of some systems can be found, for example, in [8–10]). We consider some computer algebra systems that, to some extent, can operate with tensors.

[3] Semicolon in (18) denotes covariant derivative.

3.1. Requirements for Computer Algebra Systems

Three types of tensor notation correspond to three types of tensor analytical calculations; this leads to certain requirements for computer algebra systems.

Nonindex computing handles with tensors as integral algebraic objects. In this case, one can either specify the simplest type of symmetry (the object is a representation of a group or algebra) or use objects with predefined symmetry.

Abstract indices require the ability of specifying complex types of symmetry, for example, through Young diagrams. In addition, it is necessary to be able to work with dummy indices, specify and consider them when bringing into canonical form. Both types of abstract computations use information about symmetries for bringing into canonical form and simplifying tensor expressions.

Component indices require, in fact, a scalar system of computer algebra and possibly simple matrix operations. In fact, a specific coordinate system and metric are specified. Since all operations are performed by components, the information about the tensor as an integral object and about its symmetries is lost. Therefore, all operations with symmetries and reduction to the canonical form must be carried out in the previous phase of the study.

3.2. Notation

The use of computer algebra systems often implies interactive functioning of users. In this case, the convenience of notation plays a key role. Historically, the mathematical notation of tensors follows the notation of the $\mathrm{T_EX}$ system; namely, the tensor T^a_b is written as `T^{a}_{b}`. Therefore, the use of this notation would be quite natural. This approach was implemented in *Cadabra*; however, this is a specialized system for tensor computing. A tensor notation for general-purpose computer algebra systems should account for limitations of these systems (for example, the symbol ^ is normally reserved and used for exponentiation).

Because general-purpose systems operate with functions and the basic internal data structure is a list, a functional–list notation is used. The name of a function can be given by the tensor name, and the covariant and contravariant indices are given either by a prefix (for example, as in *xAct*):

```
T(a,-b),
```

or positionally (for example, as in the *Maxima*):

```
T([a],[b]).
```

It is also possible to use associative lists, such as

```
Tensor[Name["T"], Indices[Up[a], Down[b]]].
```

Let us consider the most interesting (for practical use) implementations of tensor calculations in different computer algebra systems.

3.3. Cadabra

Cadabra (http://cadabra.phi-sci.com/) is classified among specialized computer algebra systems. The area of its specialization is the field theory. Since complex tensor calculations are an integral part of the field theory, it is not surprising that tensor calculations in this system are supported at a high level.

However, the field theory operates mostly with abstract indices, and component computations receive much less attention. Most likely, this is the reason that component computations have not yet been implemented in Cadabra, although this option has been projected for implementation.

However, component computations require that a computer algebra system possess capabilities of general-purpose systems, which cannot be found in Cadabra.

3.4. Maxima

Maxima (http://maxima.sourceforge.net/) is one of the major freeware general-purpose computer algebra systems. Maxima was derived from Macsyma, a system developed in MIT from 1968 to 1982.

Maxima implements all three types of tensor calculations [11]:

(*) package *atensor*—nonindex algebraic calculations (with a set of basic algebras and main purpose of simplifying tensor expressions by manipulations with both monoterm and multiterm symmetries);

(*) package *ctensor*—component calculations (with an option of manipulating with metrics and connectivities, and with a set of the most commonly used metrics);

(*) package *itensor*—calculations with the use of (abstract) indices.

This system duplicates the functionality of the Macsyma package and actually seeks no further development. At present, the capabilities of these packages can hardly be considered as satisfactory.

3.5. Reduce

Reduce (http://www.reduce-algebra.com/) [12] is one of the oldest (among currently existing systems) general-purpose computer algebra systems. In 2009, its license was replaced from commercial to a BSD-type. However, it should be noted that this was rather behindhand because the community at that time had dealt with other free computer algebra systems, and thus the system benefited little from the transition to a free license.

The basic system consists of:

(*) the package *atensor* mentioned above;

(*) the package *redten* (http://www.scar.utoronto.ca/~harper/redten.html) designed for component calculations.

3.6. Maple

Maple (http://www.maplesoft.com/products/Maple/index.aspx) is a commercial general-purpose computer algebra system involving also tools for numerical computations.

(*) Maple has built-in tools for manipulating with tensor components, which are not inferior to similar tools in other computer algebra systems. Actually, there are two packages:

(–) package *tensor*, which was originally designed for addressing problems of the general theory of relativity and component calculations;

(–) the powerful package *DifferentialGeometry* (which is currently a part of the main system) involves the subpackage *Tensor* designed also for component calculations. Its great advantage is the possibility to use both the tensor analysis and the whole power of differential geometry (for example, the use of symmetries of groups and Lie algebras);

(*) *GRTensor II* (http://grtensor.phy.queensu.ca/) (GPL-license) is one of the most powerful packages of component tensor calculations.

3.7. Mathematica

Mathematica (http://www.wolfram.com/mathematica/) is a commercial general-purpose computer algebra system developed by the Wolfram Research Company. The system involves a large number of computational and interactive tools (for creating mathematical textbooks).

(*) MathTensor (http://smc.vnet.net/MathTensor.html) [13] is a commercial package designed primarily for algebraic manipulations with tensors.

(*) Cartan (Tensors in Physics) (http://www.adinfinitum.no/cartan/) is a commercial package designed primarily for calculations in the theory of general relativity. Since calculations are performed in specific metrics, the package operates with component indices.

(*) Ricci (http://www.math.washington.edu/~lee/Ricci/) is a package that generally supports algebraic manipulations and has also some elements of component operations. This package is in a state of stagnation.

(*) The set of packages *xAct* (http://www.xact.es/) (GPL-license) covers operations with both abstract and component indices.

In the next section, we consider in more detail two computer algebra systems: *Cadabra* and *Maxima*.

The most important criterion for the choice of these two systems was their license: only free computer algebra systems had been considered. Thus, we excluded from consideration the extension packs to commercial computer algebra systems (regardless of the license for packages themselves).

Currently, the most advanced free computer algebra system for tensor manipulations is *Cadabra*. However, this system has not yet supported component cal-

culations. Therefore, we took system Maxima as a companion to it.

A possible candidate for consideration could be *Axiom*, but this system is currently broken into several parts that are not quite compatible.

4. TENSOR OPERATIONS IN CADABRA

To demonstrate the capabilities of Cadabra, we consider different operations over tensors in this system. Since the current version of Cadabra cannot operate with components, component operations will not be considered.

4.1. Nonindex Computing

Let us define commuting rules and check that they are satisfied:

```
{A,B}::Commuting.
{C,D}::AntiCommuting.
```

(*) The case of commuting tensors:

```
B A;
```

$$1 := BA;$$

```
@prodsort!(%);
```

$$1 := AB;$$

(*) The case of anticommuting tensor:

```
D C;
```

$$2 := DC;$$

```
@prodsort!(%);
```

$$2 := -CD;$$

4.2. Holonomic Coordinates

Let us show that, in the case of agreed metric and connectivity, the covariant derivative of the metric tensor is vanishing (see (5)).

We define a set of indices, metric, and partial derivative:

```
{a,b,c,d,e,f,g,h,i,j,
  k,l,m,n,o,p,q,r,s,t,u#}::Indices.
g_{a b}::Metric.
\partial_{#}::PartialDerivative.
```

We write the covariant derivative in the Christoffel symbols and the Christoffel symbols through the metric tensor, which just specifies the consistency of metric with connectivity:

```
\nabla := \partial_{c}{g_{a b}} -
  g_{a d}\Gamma^{d}_{b c} -
  g_{d b}\Gamma^{d}_{a c};
```

$$\nabla := \partial_c g_{ab} - g_{ad}\Gamma^d_{bc} - g_{db}\Gamma^d_{ac};$$

```
Gamma = Gamma^{a}_{b c} -> (1/2) g^{a d}
  ( \partial_{b}{g_{d c}} +
  \partial_{c}{g_{b d}} -
  \partial_{d}{g_{b c}} );
```

$$Gamma := \Gamma^a_{bc} \to \frac{1}{2} g^{ad}(\partial_b g_{dc} + \partial_c g_{bd} - \partial_d g_{bc});$$

We substitute the expression of the Christoffel symbol through the metric tensor in the expression for the covariant derivative:

```
@substitute!(\nabla)(@(Gamma));
```

$$\nabla := \partial_c g_{ab} - \frac{1}{2} g_{ad} g^{de}(\partial_b g_{ac} + \partial_c g_{be} - \partial_e g_{bc})$$

$$- \frac{1}{2} g_{db} g^{de}(\partial_a g_{ec} + \partial_c g_{ae} - \partial_e g_{ac});$$

and open the brackets:

```
@distribute!(%);
```

$$\nabla := \partial_c g_{ab} - \frac{1}{2} g_{ad} g^{de} \partial_b g_{ec} - \frac{1}{2} g_{ad} g^{de} \partial_c g_{be}$$

$$+ \frac{1}{2} g_{ad} g^{de} \partial_e g_{bc} - \frac{1}{2} g_{db} g^{de} \partial_a g_{ec}$$

$$- \frac{1}{2} g_{db} g^{de} \partial_c g_{ae} + \frac{1}{2} g_{db} g^{de} \partial_e g_{ac};$$

We raise and drop the indices unless all metric tensors are eliminated, as indicated by the double exclamation mark:

```
@eliminate_metric!!(%);
```

$$\nabla := \partial_c g_{ab} - \frac{1}{2} \partial_b g_{ac} - \frac{1}{2} \partial_c g_{ba} + \frac{1}{2} \partial_a g_{bc}$$

$$- \frac{1}{2} \partial_a g_{bc} - \frac{1}{2} \partial_c g_{ab} + \frac{1}{2} \partial_b g_{ac};$$

Then, we bring the expression into the canonical form and collect similar terms. The result is zero as expected:

```
@canonicalise!(%);
```

$$\nabla := \partial_c g_{ab} - \frac{1}{2} \partial_b g_{ac} - \frac{1}{2} \partial_c g_{ab}$$

$$+ \frac{1}{2} \partial_a g_{bc} - \frac{1}{2} \partial_a g_{bc} - \frac{1}{2} \partial_c g_{ab} + \frac{1}{2} \partial_b g_{ac};$$

```
@collect_terms!(%);
```

$$\nabla := 0;$$

4.3. γ-Matrices

Cadabra has advanced tools for operating with γ-matrices of any dimension. For definiteness, we consider γ-matrices of Dirac 4-spinors.

To simplify the calculations, we define a set of actions that are executed after each operation:

```
::PostDefaultRules( @@prodsort!(%),
  @@eliminate_kr!(%),
  @@canonicalise!(%),
  @@collect_terms!(%) ).
```

We define the indices and the values they run:

```
{a,b,c,d,e,f}::Indices(vector).
{a,b,c,d,e,f}::Integer(0..3).
```

The space dimension will be used for finding the track of the Kronecker delta.

The γ-matrices are specified using the metric (see (7)):

```
\gamma_{#}::GammaMatrix(metric=g).
g_{a b}::Metric.
g_{a}^{b}::KroneckerDelta.
```

Now, we demonstrate some symmetry identities that γ-matrices satisfy.

(*) Clifford-Dirac equation (7):

```
\gamma{a} \gamma_{b} +
  \gamma_{b} \gamma_{a};
```

$$1 := \gamma_a\gamma_b + \gamma_b\gamma_a;$$

The algorithm `@join` converts pairwise products of γ-matrices into sums of γ-matrices of higher valences (see (10)). The additional argument `expand` indicates that the rules of antisymmetrization for γ-matrices are taken into account:

```
@join!(%){expand};
```

$$1 := 2g_{ab};$$

(*) Convolution of two γ-matrices (12):

```
\gamma^{a} \gamma_{a};
```

$$2 := \gamma^a\gamma_a;$$

```
@join!(%){expand};
```

$$2 := 3;$$

(*) identity (13):

```
\gamma^{a} \gamma^{b}
  \gamma^{c} \gamma_{a};
```

$$3 := \gamma^a\gamma^b\gamma^c\gamma_a;$$

```
@join!(%){expand};
```

$$3 := (\gamma^{ab} + g^{ab})(-\gamma_a^c + g_a^c);$$

```
@distribute!(%){expand};
```

$$3 := -\gamma^{ba}{}_a^c - 2\gamma^{bc} + g^{bc};$$

```
@join!!(%){expand};
```

$$3 := 4g^{bc};$$

(*) identity (14):

```
\gamma_{a} \gamma_{b};
```

$$4 := \gamma_a\gamma_b;$$

```
@join!!(%){expand};
```

$$4 := \gamma_{ab} + g_{ab};$$

(*) identity (15):

```
\gamma_{a} \gamma_{b} \gamma_{c};
```

$$5 := \gamma_a\gamma_b\gamma_c;$$

```
@join!!(%){expand};
```

$$5 := (\gamma_{ab} + g_{ab})\gamma_c;$$

```
@distribute!(%);
```

$$5 := \gamma_{ab}\gamma_c + \gamma_c g_{ab};$$

```
@join!!(%){expand};
```

$$5 := \gamma_{abc} + \gamma_a g_{bc} - \gamma_b g_{ac} + \gamma_c g_{ab};$$

(*) the more complex identity

$$\gamma_{ab}\gamma_{bc}\gamma_{de} = -4\gamma_{cd} + 21g_{cd}\hat{I}. \qquad (19)$$

in Cadabra has the following form:

```
\gamma_{a b} \gamma_{b c}
  \gamma_{d e} \gamma_{e a};
```

$$6 := -\gamma_{ab}\gamma_{ca}\gamma_{de}\gamma_{be};$$

```
@join!!(%){expand};
```

$$6 := -(2\gamma_{bc} + 3g_{bc})(2\gamma_{bd} - 3g_{bd});$$

```
@distribute!(%);
```

$$6 := -4\gamma_{cb}\gamma_{db} - 12\gamma_{cd} + 9g_{cd};$$

```
@join!(%){expand};
```

$$6 := -4\gamma_{cd} + 21g_{cd};$$

4.4. Monoterm Symmetries

As an example of monoterm symmetry, we consider the symmetries of Riemann tensor (16). To this end, we first specify the symmetry properties using Young diagrams:

```
R_{a     b     c     d}::TableauSymme-
try(shape={2,2},
  indices={0,2,1,3}).
```

In this example, the symmetry has the form $\begin{array}{|c|c|}\hline a & c \\ \hline b & d \\ \hline \end{array}$.

Then, we perform symmetric and antisymmetric permutation of the indices. The algorithm `@canonicalise` brings the operand into the canonical form, accounting for monoterm symmetries.

```
R_{c d a b};
```

$$1 := R_{cdab};$$

```
@canonicalise!(%);
```

$$1 := R_{abcd};$$

```
R_{a b c d} + R_{b a c d};
```

$$2 := R_{abcd} + R_{bacd};$$

```
@canonicalise!(%);
```

$$2 := R_{abcd} - R_{bacd};$$

```
@collect_terms!(%);
```

$$2 := 0;$$

4.5. Multiterm Symmetries

We demonstrate the operation with multiterm symmetries by an example of the Riemann tensor.

We introduce notations for indices and derivatives:

```
{a,b,c,d,e,f,g#}::Indices(vector).
\nabla{#}::Derivative.
```

The symmetry can be specified by the Young $\begin{array}{|c|c|c|}\hline a & c & e \\ \hline b & d \\ \cline{1-2} \end{array}$ diagram, as in the previous case:

```
\nabla_{e}{R_{a  b  c  d}}::TableauSymme-
try(
  shape={3,2}, indices={1,3,0,2,4}).
```

However, it is more convenient to use the following notation:

```
  R_{a b c d}::RiemannTensor.
```

Similarly, we deal with the covariant derivative of the Riemann tensor, which satisfies the differential Bianchi identity:

```
\nabla_{e}{R_{a  b  c  d}}::SatifiesBian-
chi.
```

Let us check first Bianchi identity (17).

```
R_{a b c d} + R_{a c d b} + R_{a d b c};
```

$$1 := R_{abcd} + R_{acdb} + R_{adbc};$$

```
@young_project_tensor!2(%){ModuloMono-
term}:
@collect_terms!(%);
```

$$1 := 0;$$

Now, we demonstrate that the second (differential) Bianchi identity (18) is satisfied:

```
\nabla_{e}{R_{a b c d}} +
  \nabla_{e}{R_{a b d e}} +
  \nabla_{e}{R_{a d b c}};
```

$$2 := \nabla_e R_{abcd} + \nabla_c R_{abde} + \nabla_d R_{abec};$$

```
@young_project_tensor!2(%){ModuloMono-
term}:
@collect_terms!(%);
```

$$2 := 0;$$

5. AN EXAMPLE OF TENSOR CALCULATIONS IN MAXIMA

Since Cadabra currently does not support component computations, we demonstrate them in the Maxima system. As an example, we consider the Maxwell equations written in the cylindrical coordinates in a holonomic basis [1].

First, we load a small package written by us that contains definitions for differential operators:

```
(%i1) load("diffop.mac|)$.
```

We define a cylindrical coordinate system:

```
(%i2) ct_coordsys(polarcylindrical)$ .
```

We consider the components of the metric tensor g_{ij}:

```
(%i3) lg.
```

$$(\%o3)\begin{pmatrix} 1 & 0 & 0 \\ 0 & r^2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

We calculate and consider the components of the metric tensor g_{ij}:

```
(%i4) cmetric()$.
(%i5) ug;
```

$$(\%o5)\begin{pmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{r^2} & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

We define the required vectors and determine their coordinate dependences. Due to the limitations of Maxima, we denote j^1 as `j1` and j_1 as `j_1`:

```
(%i6) j:[j1,j2,j3]$
  depends(j,cons(t, ct_coords))$
(%i8) B:[B1,B2,B3]$
  depends(B,cons(t, ct_coords))$
(%i10) D:[D1,D2,D3]$
  depends(D,cons(t, ct_coords))$
(%i12) H:[H_1,H_2,H_3]$
  depends(H,cons(t, ct_coords))$
(%i14) E:[E_1,E_2,E_3]$
  depends(E,cons(t, ct_coords))$
```

Now, we calculate all sides of the Maxwell equations written in the cylindrical coordinates:

```
(%i16) Div(B);
```

$$(\%o16)\ \frac{d}{dz}B3 + \frac{d}{d\theta}B2 + \frac{d}{dr}B1 + \frac{B1}{r}$$

```
(%i17) Div(D) -4*%pi*rho;
```

$$(\%o17)\ \frac{d}{dz}D3 + \frac{d}{d\theta}D2 + \frac{d}{dr}D1 + \frac{D1}{r} - 4\pi\rho$$

```
(%i18) Rot(H)
    + diff(transpose(matrix(D)),t)/c
    - 4*%pi/c*transpose(matrix(j));
```

$$(\%o18)\begin{pmatrix} \frac{\frac{d}{d\theta}H\_3 - \frac{d}{dz}H\_2}{|r|} + \frac{\frac{d}{dt}D1}{c} - \frac{4\pi j1}{c} \\ -\frac{\frac{d}{dr}H\_3 - \frac{d}{dz}H\_1}{|r|} + \frac{\frac{d}{dt}D2}{c} - \frac{4\pi j2}{c} \\ \frac{\frac{d}{dr}H\_2 - \frac{d}{d\theta}H\_1}{|r|} + \frac{\frac{d}{dt}D3}{c} - \frac{4\pi j3}{c} \end{pmatrix}$$

```
(%i19) Rot(E)
    + diff(transpose(matrix(D)),t)/c;
```

$$(\%o19)\begin{pmatrix} \frac{\frac{d}{d\theta}E\_3 - \frac{d}{dz}E\_2}{|r|} + \frac{\frac{d}{dt}B1}{c} \\ \frac{\frac{d}{dt}B2}{c} + \frac{\frac{d}{dr}E\_3 - \frac{d}{dz}E\_1}{|r|} \\ \frac{\frac{d}{dr}E\_2 - \frac{d}{d\theta}E\_1}{|r|} + \frac{\frac{d}{dt}B3}{c} \end{pmatrix}$$

Thus, the result coincided with the analytical expressions obtained in [1].

6. CONCLUSIONS

This paper originated from the desire of the authors to conduct bulky tensor calculations in a computer algebra system. The search for an appropriate system made us formulate a set of criteria that such a system should satisfy.

Because there are several types of tensor calculations, the full implementation of tensor calculations in computer algebra systems requires a broad set of capabilities. Unfortunately, at present, there are almost no systems with a fully satisfactory support of tensors.

Component tensor calculations require almost no additional capabilities that are not available in a universal computer algebra system. Therefore, the packages implementing this functionality are used most

widely (for example, *Maxima*, *Maple*, and *Mathematica*).

The tensor notation is very different from the usual functional notation that is used by the vast majority of computer algebra systems. Therefore, the efficient operation with tensors would require a specialized system (or a specialized add-on over a universal system) that supports the natural tensor notation (for example, *Cadabra*).

As a result, we failed to find a system that fully meets the needs of tensor calculus. At the moment, the authors use the *Cadabra* specialized system and a universal computer algebra system (currently, *Maxima*).

REFERENCES

1. Kulyabov, D.S., Korolkova, A.V., and Korolkov, V.I., Maxwell's equations in arbitrary coordinate system, *Bull. PFUR. Ser. Math. Inf. Sci. Phys.*, 2012, no. 1, pp. 96–106.
2. Kulyabov, D.S. and Nemchaninova, N.A., Maxwell's equations in curvilinear coordinates, *Bull. PFUR. Ser. Math. Inf. Sci. Phys.*, 2011, no. 2, pp. 172–179.
3. Sardanishvili, G.A., *Sovremennye metody teorii polja* (Modern Methods of the Field Theory), vol. 2: *Geometriya i klassicheskaya mekhanika* (Geometry and Classical Mechanics), Moscow: URSS, 1998.
4. Cartan, E., *The Theory of Spinors*, Paris: Hermann, 1966.
5. Gerdt, V.P., Tarasov, O.V., and Shirkov, D.V., Analytical calculations on digital computers for applications in physics and mathematics, *Sov. Phys. Usp.*, 1980, vol. 130, no. 1, pp. 59–78. http://ufn.ru/ru/articles/1980/1/d/
6. Barut, A. and Raczka, R., *Theory of Group Representations and Applications* Warsaw: Polish Scientific Publishers, 1980.
7. Penrose, R. and Rindler, W., *Spinors and Space–Time. Two-Spinor Calculus and Relativistic Fields*, Cambridge Univ. Press, 1987, vol. 1.
8. Computer Algebra Information Network. http://www.computeralgebra.nl/systemsoverview/special/systems.html
9. Kulyabov, D.S. and Kokotchikova, M.G., Analytical review of symbolic computing systems, *Bull. PFUR. Ser. Math. Inf. Sci. Phys.*, 2007, nos. 1–2, pp. 38–45.
10. Sevastianov, L.A., Kulyabov, D.S., and Kokotchikova, M.G., An application of computer algebra system Cadabra to scientific problems of physics, *Phys. Part. Nucl. Lett.*, 2009, no. 6, no. 7, pp. 530–534.
11. Toth, V., Tensor manipulation in GPL Maxima, 2005. arXiv:cs/0503073v2
12. Gerdt, V.P. and Tiller, P., *A reduce program for symbolic computation of Puiseux expansions,* Dubna: Joint Inst. Nucl. Res., 1991.
13. Parker, L. and Christensen, S.M., *MathTensor: A System for Doing Tensor Analysis by Computer*, Addison-Wesley, 1994.

Translated by V. Arutyunyan