

Tensor computations in computer algebra systems

A. V. Korolkova,* D. S. Kulyabov,† and L. A. Sevastyanov‡
Telecommunication System Department
Peoples' Friendship University of Russia
Miklukho-Maklaya str., 6, Moscow, 117198, Russia§

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.

I. 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 also important in the case of tensor calculus.

II. MAIN APPLICATION FIELDS AND TYPES OF TENSOR NOTATIONS

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

A. Nonindex Computations for Theoretical Constructs

Nonindex computations are commonly used in theoretical constructs and often opposed to component computations. We implement the main 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 $\begin{bmatrix} p \\ q \end{bmatrix}$:

$$A + B = C.$$

The addition of tensors gives 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 $\begin{bmatrix} p+r \\ q+s \end{bmatrix}$:

$$A \otimes D = E.$$

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 last indices by \mathfrak{C} . 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 $\begin{bmatrix} p \\ q \end{bmatrix}$:

$$\mathfrak{C}F = G.$$

- *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).

B. Vector Computations

Vector calculus is the simplest case of tensor calculus (a vector is a tensor of valence one). The N -dimensional 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) [7, 8].

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

* akorolkova@sci.pfu.edu.ru

† yamadharma@gmail.com

‡ leonid.sevast@gmail.com

§ Published in: A. V. Korol'kova, D. S. Kulyabov, and L. A. Sevast'yanov. Tensor computations in computer algebra systems. *Programming and Computer Software*, 39(3):135–142, 2013. ISSN 0361-7688. doi: 10.1134/S0361768813030031.; Sources: https://bitbucket.org/yamadharma/articles-2011-cas_tensor

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:

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

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

$$\nabla_k g_{ij} = 0. \quad (1)$$

In this case, the connectivity and metric are consistent [9].

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 the dimension unchanged under a coordinate transformation¹ (for details, see [8]):

$$\vec{\delta}_i = \frac{\partial}{\partial s^i}, \quad \vec{\delta}^i = ds^i, \quad ds^i = h_j^i dx^j.$$

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

C. 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 $\frac{1}{2}$. 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 [3]:

$$\gamma(a\gamma b) = g_{ab}\hat{I}, \quad (2)$$

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:

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

where 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 spin to vector space, the spin coefficients should be introduced more strictly and Eq. (2) should be written in the following way

$$\gamma_{a\rho}^\sigma \gamma_{b\sigma}^\tau + \gamma_{b\rho}^\sigma \gamma_{a\sigma}^\tau = 2g_{ab}\delta_\rho^\sigma.$$

The construction of a complete Clifford algebra requires also products of γ -matrices, but in view of (2), it is suffice to consider only antisymmetrized products

$$\gamma_{ab\dots d} := \gamma_{[a}\gamma_b \cdots \gamma_{d]}. \quad (3)$$

Also, we introduce an element γ_5 :

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

where e^{abcd} is an alternating tensor.

The handling with γ -matrices is reduced to a set of relations following from algebraic symmetries, such as

$$\gamma^a \gamma_a = 4\hat{I}, \quad (4)$$

$$\gamma^a \gamma^b \gamma^c \gamma_a = 4g^{bc}\hat{I}, \quad (5)$$

$$\gamma_a \gamma_b = \gamma_{ab} + g_{ab}\hat{I}, \quad (6)$$

$$\gamma_a \gamma_b \gamma_c = \gamma_{abc} + g_{ab}\gamma_c + g_{bc}\gamma_a - g_{ac}\gamma_b. \quad (7)$$

D. Tensor Calculations in the Theory of General Relativity

General relativity became the first physical theory requiring the whole 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 monoterms and multiterms symmetries. A key element of the theory is the Riemann tensor, which has both very simple monoterms and complex multiterms symmetries like the Bianchi identities.

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

$$R_{bacd} = -R_{abcd}, \quad R_{cdab} = R_{abcd}. \quad (8)$$

Multiterms symmetries are given by an algebra of permutations. The Bianchi identity has the form²:

$$R_{a(bc d)} = R_{abcd} + R_{acdb} + R_{adbc} = 0. \quad (9)$$

The differential (second) Bianchi identity has the form³:

$$R_{ab(cd;e)} = \nabla_e R_{abcd} + \nabla_c R_{abde} + \nabla_d R_{abec} = 0. \quad (10)$$

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

² The round brackets in (9) denote symmetrization.

³ Semicolon in (10) denotes covariant derivative.

It is most reasonable to specify symmetries by means of Young diagrams [2]. 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 $\begin{array}{|c|c|} \hline a & b \\ \hline c & d \\ \hline \end{array}$.

E. 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 non-index 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 in its 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 notations or add verbal explanations. It is this problem that should be treated by abstract indices [11].

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 obeying 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.

III. TENSOR COMPUTATIONS AND COMPUTER ALGEBRA SYSTEMS

Modern computer algebra systems are able to 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 [1, 6, 12]). We consider some computer algebra systems that to some extent can operate with tensors.

A. Requirements for Computer Algebra System

Three types of tensor notations 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 the presence of simple matrix operations. In fact, a specific coordinate system and metric are given. 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 bringing into canonical form must be carried out in the previous phase of the study.

B. 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 \TeX system; namely, the tensor T_b^a 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 the limitations of these systems (for example, the symbol \sim 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.

C. Cadabra

Cadabra [<http://cadabra.phi-sci.com/>] refers to the type of specialized computer algebra systems. The

area of its specialization is the field theory. Because 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 the capabilities of general-purpose systems, which cannot be found in Cadabra.

D. 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 the three types of tensor calculations [13]:

- package *atensor*—nonindex algebraic calculations (with a set of basic algebras and main purpose of simplifying tensor expressions by manipulations with both monoterms 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 by using (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.

E. Reduce

Reduce [<http://www.reduce-algebra.com/>] [4] 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.

F. Maple

Maple [<http://www.maplesoft.com/products/Maple/index.aspx>] is a commercial general-purpose computer algebra system involving also the 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 has currently been involved in 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.

G. 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>] [10] 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 the 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 calculations. Therefore, we took the 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.

IV. 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.

A. 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!(%);
```

$$2 := -CD;$$

B. Holonomic Coordinates

Let us show that in the case of agreed metric and connectivity, the covariant derivative of the metric tensor is vanishing (see (1)).

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};
```

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

```
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}} );
```

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

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

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

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

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!!(%);
```

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

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

`@canonicalise!(%)`;

$$\begin{aligned} \nabla := & \partial_c g_{ab} - \frac{1}{2} \partial_b g_{ac} - \frac{1}{2} \partial_c g_{ab} + \\ & + \frac{1}{2} \partial_a g_{bc} - \frac{1}{2} \partial_a g_{bc} - \frac{1}{2} \partial_c g_{ab} + \frac{1}{2} \partial_b g_{ac}; \end{aligned}$$

`@collect_terms!(%)`;

$$\nabla := 0;$$

C. 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 their running values:

```
{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 (2)):

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

Now, we demonstrate some symmetry identities satisfied by γ -matrices.

- Clifford-Dirac equation (2):

$$\begin{aligned} & \gamma_a \gamma_b + \\ & \gamma_b \gamma_a \end{aligned}$$

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

The algorithm `@join` converts the pairwise products of γ -matrices into the sum of γ -matrices of higher valences (see (3)). The additional argument `expand` indicates that the rules of antisymmetrization for γ -matrices are taken into account:

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

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

- Convolution of two γ -matrices (4):

`\gamma^a \gamma_a`;

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

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

$$2 := 4;$$

- Identity (5):

`\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!(%)`;

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

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

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

- Identity (6):

`\gamma_a \gamma_b`;

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

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

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

- Identity (7):

`\gamma_a \gamma_b \gamma_c`;

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

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

```

5 := (\gamma_{ab} + g_{ab})\gamma_c;
@distributed!(%);

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};

```

- A more complex identity:

$$\gamma_{ab}\gamma_{bc}\gamma_{de}\gamma_{ea} = -4\gamma_{cd} + 21g_{cd}\hat{I}$$

In Cadabra, this identity has the form

```

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

```

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

```

@join!!(%){expand};

```

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

```

@distributed!(%);

```

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

```

@join!(%){expand};

```

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

D. Monoterm Symmetries

As an example of monoterm symmetry, we consider the symmetries of Riemann tensor (8). To this end, we first specify the symmetry properties using Young diagrams:

```

R_{a b c d}::TableauSymmetry(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!(%);

```

$$2 := 0;$$

E. 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 diagram

$$\begin{array}{|c|c|c|} \hline a & c & e \\ \hline b & d & \\ \hline \end{array}$$

, as in the previous case:

```

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

```

However, it is more convenient 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}}::SatisfiesBianchi.

```

Let us check first Bianchi identity (9).

```

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

```

$$1 := R_{abcd} + R_{acdb} + R_{adb c};$$

```

@young_project_tensor!2(%){ModuloMonoterm}:

```

```

@collect_terms!(%);

```

$$1 := 0;$$

Now, we demonstrate that second (differential) Bianchi identity (10) is satisfied:

```

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

```

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

```

@young_project_tensor!2(%){ModuloMonoterm}:

```

```

@collect_terms!(%);

```

$$2 := 0;$$

V. AN EXAMPLE OF TENSOR CALCULATIONS IN MAXIMA

Because Cadabra currently does not support component computations, we demonstrate them in the Maxima system. As an example, we consider the Maxwell equations written in cylindrical coordinates in a holonomic basis [8].

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(polar cylindrical)$
```

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 by j1 and j_1 by 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 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) 
$$\begin{pmatrix} \frac{\frac{d}{d\theta} H}{|r|} \frac{3 - \frac{d}{dz} H}{|r|} \frac{2}{c} + \frac{\frac{d}{dt} D1}{c} - \frac{4\pi j1}{c} \\ -\frac{\frac{d}{dr} H}{|r|} \frac{3 - \frac{d}{dz} H}{|r|} \frac{1}{c} + \frac{\frac{d}{dt} D2}{c} - \frac{4\pi j2}{c} \\ \frac{\frac{d}{dr} H}{|r|} \frac{2 - \frac{d}{d\theta} H}{|r|} \frac{1}{c} + \frac{\frac{d}{dt} D3}{c} - \frac{4\pi j3}{c} \end{pmatrix}$$

```

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

```
(%o19) 
$$\begin{pmatrix} \frac{\frac{d}{d\theta} B2}{|r|} \frac{3 - \frac{d}{dz} E}{|r|} \frac{2}{c} + \frac{\frac{d}{dt} B1}{c} \\ \frac{\frac{d}{dr} B2}{|r|} \frac{3 - \frac{d}{dz} E}{|r|} \frac{1}{c} \\ \frac{\frac{d}{dr} E}{|r|} \frac{2 - \frac{d}{d\theta} E}{|r|} \frac{1}{c} + \frac{\frac{d}{dt} B3}{c} \end{pmatrix}$$

```

Thus, the result coincided with the analytical expressions obtained in [8].

VI. CONCLUSIONS

This paper was prompted by the fact that the authors intended to conduct bulky tensor calculations in computer algebra systems. The search for an appropriate system brought them to formulate a set of criteria that must be met by such a computer algebra system.

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 features of the 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 the 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*).

-
- [1] Computer Algebra Information Network. URL <http://www.computeralgebra.nl/systemoverview/special/systems.html>.
- [2] A. Barut and R. Rączka. *Theory of Group Representations and Applications*. World Scientific, 1986. ISBN 9789971502171.
- [3] E. Cartan. *The Theory of Spinors*. Paris: Hermann, 1966.
- [4] V. P. Gerdt and P. Tiller. *A Reduce program for symbolic*

computation of Puiseux expansions. Joint Inst. Nuclear Res., Dubna, 1991.

- [5] V. P. Gerdt, O. V. Tarasov, and D. V. Shirkov. Analytic calculations on digital computers for applications in physics and mathematics. *Physics-Uspekhi*, 23(1):59–77, 1980. doi:10.1070/PU1980v023n01ABEH004898. URL <http://ufn.ru/en/articles/1980/1/d/>. in Russian.
- [6] D. S. Kulyabov and M. G. Kokotchkova. Analytic Review of Computer Algebra System. *Bulletin of PFUR*.

- Series "Mathematics. Information Sciences. Physics", (1–2):38–45, 2007. in Russian.*
- [7] D. S. Kulyabov and N. A. Nemchaninova. Maxwell's Equations in Curvilinear Coordinates. *Bulletin of Peoples' Friendship University of Russia. Series Mathematics. Information Sciences. Physics*, (2):172–179, 2011. in Russian.
- [8] D. S. Kulyabov, A. V. Korolkova, and V. I. Korolkov. Maxwell's Equations in Arbitrary Coordinate System. *Bulletin of PFUR. Series "Mathematics. Information Sciences. Physics", (1):96–106, 2012.*
- [9] L. Mangiarotti and G. A. Sardanashvili. *Connections in Classical and Quantum Field Theory*. World Scientific, 2000. ISBN 9789812813749.
- [10] L. Parker and S. M. Christensen. *MathTensor: a system for doing tensor analysis by computer*. Addison-Wesley, 1994.
- [11] R. Penrose and W. Rindler. *Spinors and Space-Time. Two-Spinor Calculus and Relativistic Fields*, volume 1. Cambridge University Press, 1987.
- [12] L. A. Sevastianov, D. S. Kulyabov, and M. G. Kokotchkova. An Application of Computer Algebra System Cadabra to Scientific Problems of Physics. *Particles and Nuclei, Letters*, 6(7 (156)):39–44, 2009. in Russian.
- [13] V. Toth. Tensor Manipulation in GPL Maxima, 2005. URL <http://arxiv.org/abs/cs/0503073>.

Тензорные расчёты в системах компьютерной алгебры

А. В. Королькова,* Д. С. Кулябов,† and Л. А. Севастьянов‡

Кафедра систем телекоммуникаций
Российский университет дружбы народов
ул. Миклухо-Маклая, д.6, Москва, 117198, Россия§

В статье рассмотрены три вида тензорных расчётов. В соответствии с ними авторы попытались сформулировать критерии, которым должна удовлетворять система компьютерной алгебры для работы с тензорами. Сделан краткий обзор текущего состояния тензорных вычислений в разных системах компьютерной алгебры. Тензорные расчёты проиллюстрированы соответствующими примерами, реализованными в конкретных системах: Cadabra и Maxima.

I. ВВЕДЕНИЕ

Тензорные вычисления используются во многих областях физики. Следует заметить, что во всей своей мощи формализм тензорного анализа проявляется не во всех областях, достаточно часто используют его упрощённые варианты.

Каждая тензорная операция сама по себе достаточно проста. Однако даже при стандартных вычислениях приходится выполнять множество элементарных операций. Эти операции требуют большой внимательности и скрупулёзности. Именно поэтому в данной области актуальны разные упрощения нотации, оптимизация операций (например, тензорные диаграммы Пенроуза).

Одной из задач систем компьютерной алгебры является освобождение исследователя от рутинных операций, что актуально и в случае тензорного исчисления.

II. ОСНОВНЫЕ ОБЛАСТИ ПРИМЕНЕНИЯ И ТИПЫ ЗАПИСИ ТЕНЗОРОВ

Чтобы определить основные виды операции с тензорами, рассмотрим основные области их применения.

A. Безындексные вычисления для теоретических построений

Безындексные вычисления обычно применяются в теоретических построениях и часто противопоставляются компонентным вычислениям. Посмотрим, как можно реализовать основные тензорные операции в безындексном случае.

- *Сложение тензоров.* При сложении двух тензоров валентности $\begin{bmatrix} p \\ q \end{bmatrix}$ получаем тензор валентности $\begin{bmatrix} p \\ q \end{bmatrix}$:

$$A + B = C.$$

Сложение тензоров задаёт структуру абелевой группы.

- *Тензорное умножение.* При тензорном умножении тензора A с валентностью $\begin{bmatrix} p \\ q \end{bmatrix}$ на тензор D с валентностью $\begin{bmatrix} r \\ s \end{bmatrix}$ получаем тензор E с валентностью $\begin{bmatrix} p+r \\ q+s \end{bmatrix}$:

$$A \otimes D = E.$$

Тензорное умножение задаёт структуру некоммутативной полугруппы.

- *Операция свёртывания.* Обозначим операцию свёртывания тензоров по последним индексам через \mathfrak{C} . Тогда под действием этой операции тензор F с валентностью $\begin{bmatrix} p+1 \\ q+1 \end{bmatrix}$ переходит в тензор G с валентностью $\begin{bmatrix} p \\ q \end{bmatrix}$:

$$\mathfrak{C}F = G.$$

- *Операция перестановки индексов.* Данная операция необходима для задания симметрии тензоров (например, коммутатора или антикоммутатора тензора), для расширения операции свёртывания на свёртку по произвольным индексам. Однако в рамках безындексного подхода обозначить эту операцию нельзя. Впрочем, простейшие симметрии мы можем явно указать в описании объекта (при этом для однозначности придётся наложить ограничения на валентность).

B. Векторные вычисления

Векторное исчисление — простейший вариант тензорного исчисления (вектор — тензор валентности один). Вектор a^n размерности N представляется как совокупность набора компонент $n = \overline{1, N}$, зависящего от базиса, и линейного закона преобразования

* akorolkova@sci.pfu.edu.ru

† yamadharma@gmail.com

‡ leonid.sevast@gmail.com

§ Опубликовано в: А. В. Королькова, Д. С. Кулябов, and Л. А. Севастьянов. Tensor computations in computer algebra systems. *Programming and Computer Software*, 39(3):135–142, 2013. ISSN 0361-7688. doi: 10.1134/S0361768813030031.; Исходные тексты: https://bitbucket.org/yamadharma/articles-2011-cas_tensor

компонент при изменении базиса. Часто используемые операции — построение разнообразных дифференциальных операторов и замена базиса. Наиболее распространённые операторы: градиент, дивергенция, ротор (специфична для трёхмерного пространства \mathbb{R}^3) [1, 2]).

Для компонентных расчётов необходимо определить базис, метрику и связность (а соответственно и ковариантную производную). В векторном исчислении широкое распространение получил голономный базис, который строится как совокупность частных производных от координат в касательном расслоении и дуального базиса как 1-формы в кокасательном расслоении:

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

Связность и метрика строятся таким образом, чтобы ковариантная производная от метрики равнялась нулю:

$$\nabla_k g_{ij} = 0. \quad (1)$$

В этом случае связность и метрика согласованы [3].

Следует заметить, что в векторных вычислениях также часто используется специальный неголономный базис, позволяющий не различать контравариантные и ковариантные вектора, сохраняя размерность при замене координат¹ (подробнее см. в работе [1]):

$$\bar{\delta}_i = \frac{\partial}{\partial s^i}, \quad \bar{\delta}^i = ds^i, \quad ds^i = h_j^i dx^j.$$

Здесь ds^i — элемент длины по соответствующей координате, h_j^i — коэффициенты неголономности (в случае ортогональных координат — коэффициенты Ламе).

С. Дираковские 4-спиноры

Специальным случаем тензорных объектов являются спиноры (называемые также спин-тензорами). В частности спиноры являются представлениями группы Лоренца с полуцелым старшим весом. Обычные тензоры являются представлениями с целочисленным старшим весом.

По историческим причинам наиболее часто в исследованиях используются дираковские 4-спиноры, которые применяют для записи уравнений Дирака, описывающих фермионы со спином $\frac{1}{2}$. Дираковские 4-спиноры суть неприводимые спиноры для случая $n = 4$ и $s = \pm 2$, где n — размерность векторного пространства, $s = n - 2u$ — его сигнатура, u — число отрицательных значений диагонального метрического тензора g_{ab} .

Обычно для манипуляции с дираковскими спинорами используют γ -матрицы, получаемые из уравнения Клиффорда–Дирака [4]:

$$\gamma_{(a}\gamma_{b)} = g_{ab}\hat{I}, \quad (2)$$

где γ_a — матрицы $N \times N$, g_{ab} — метрический тензор, \hat{I} — единичная матрица $N \times N$, N — размерность спинорного пространства:

$$N = \begin{cases} 2^{n/2}, & \text{чётное } n, \\ 2^{n/2-1/2}, & \text{нечётное } n. \end{cases}$$

γ -матрицы являются элементами алгебры Клиффорда, порождающими линейное преобразование спинорного пространства.

Поскольку γ -матрицы можно рассматривать как коэффициенты перехода от спинорного пространства к векторному, то более строго следует ввести спинорные коэффициенты и записать уравнение (2) следующим образом:

$$\gamma_{a\rho}^\sigma \gamma_{b\sigma}^\tau + \gamma_{b\rho}^\sigma \gamma_{a\sigma}^\tau = 2g_{ab}\delta_\rho^\sigma.$$

Для построения полной алгебры Клиффорда необходимы ещё и произведения γ -матриц, однако в силу (2) достаточно рассматривать только антисимметризованные произведения:

$$\gamma_{ab\dots d} := \gamma_{[a}\gamma_b \dots \gamma_{d]}. \quad (3)$$

Также вводится элемент γ_5 :

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

где e^{abcd} — альтернирующий тензор.

Манипуляции с γ -матрицами сводятся к набору соотношений, следующих из алгебраических симметрий, например

$$\gamma^a \gamma_a = 4\hat{I}, \quad (4)$$

$$\gamma^a \gamma^b \gamma^c \gamma_a = 4g^{bc}\hat{I}, \quad (5)$$

$$\gamma_a \gamma_b = \gamma_{ab} + g_{ab}\hat{I}, \quad (6)$$

$$\gamma_a \gamma_b \gamma_c = \gamma_{abc} + g_{ab}\gamma_c + g_{bc}\gamma_a - g_{ac}\gamma_b. \quad (7)$$

Д. Тензорные вычисления в общей теории относительности

Общая теория относительности стала первой физической теорией, потребовавшей всю мощь дифференциальной геометрии и тензорных вычислений [5]. В вычислениях возникают громоздкие тензорные конструкции, которые можно упрощать, учитывая симметрии тензоров. Обычно выделяют одноэлементные (monoterm) и многоэлементные (multiterm) симметрии. Одним из основных элементов теории является тензор Римана, обладающий как простейшими одноэлементными, так и сложными многоэлементными симметриями типа тождеств Бьянки.

¹ При преобразовании длина переходит в длину, угол в угол и т.д.

Одноэлементные симметрии соответствуют простым перестановочным симметриям и задаются группой перестановок. Для тензора Римана, например, имеем:

$$R_{bacd} = -R_{abcd}, \quad R_{cdab} = R_{abcd}. \quad (8)$$

Многоэлементные симметрии задаются алгеброй перестановок. Тождество Бьянки имеет вид²:

$$R_{a(bc)d} = R_{abcd} + R_{acdb} + R_{adb c} = 0. \quad (9)$$

Дифференциальное (второе) тождество Бьянки имеет вид³:

$$R_{ab(cd;e)} = \nabla_e R_{abcd} + \nabla_c R_{abde} + \nabla_d R_{abec} = 0. \quad (10)$$

Симметрии наиболее естественно задавать с помощью диаграмм Юнга [6]. Причём наличие предопределённых классов тензоров не отменяет необходимость в явном задании симметрии. Например, тензор Римана R_{abcd} в разных источниках имеет симметрии $\begin{array}{|c|c|} \hline a & c \\ \hline b & d \\ \hline \end{array}$ или $\begin{array}{|c|c|} \hline a & b \\ \hline c & d \\ \hline \end{array}$.

Е. Типы записи тензоров

Таким образом, опираясь на рассмотренные выше виды тензорных вычислений, можно выделить три типа записи тензоров: компонентная запись, запись с абстрактными индексами и безындексная запись. Каждый тип имеет свою специфику и область применения.

Компонентные индексы, фактически, превращают тензор в набор скалярных величин, применяемых при конкретных расчётах. Обычно оперировать с компонентными индексами есть смысл лишь после упрощения тензорного выражения и учёта всех его симметрий.

Безындексную запись часто используют, если исследователя интересует не конечный результат, а симметрия тензоров. Однако эта форма записи страдает недостатком выразительности: тензор рассматривается как целостный объект, соответственно и симметрии возможно рассматривать лишь те, которые относятся к тензору в целом. Для работы с объектами сложной структуры приходится изобретать новые обозначения либо добавлять словесные пояснения. Эту проблему и должны снять абстрактные индексы [7].

Абстрактные индексы следует рассматривать как усовершенствование безындексной записи тензора. Абстрактный индекс обозначает лишь принадлежность тензора к определённому пространству, а не следование тензорному правилу преобразования (в отличие от компонентных индексов). В этом случае возможно рассмотрение как симметрий, охватывающих

весь тензор (все его индексы), так и симметрий отдельных групп индексов.

III. ТЕНЗОРНЫЕ ВЫЧИСЛЕНИЯ И СИСТЕМЫ КОМПЬЮТЕРНОЙ АЛГЕБРЫ

Современные системы компьютерной алгебры способны решать задачи достаточно широкого спектра и из разных областей знаний. Есть системы как узко специализированные, так и с претензией на универсальность (обзоры некоторых систем см., например, в работах [8–10]). Рассмотрим некоторые системы компьютерной алгебры, в которых в той или иной степени реализована возможность работы с тензорами.

А. Требования к системе компьютерной алгебры

Три типа записи тензоров соответствуют трём видам тензорных аналитических вычислений, что приводит к определённым требованиям, предъявляемым системе компьютерной алгебры.

Безындексные вычисления манипулируют с тензорами как с целостными алгебраическими объектами. В данном случае возможно либо задание самого простейшего типа симметрии (объект является представлением какой-либо группы или алгебры), либо использование объектов с заранее заданной симметрией.

Абстрактные индексы требуют возможности задания сложных типов симметрии, например, через диаграммы Юнга. Кроме того, необходимо уметь работать с неммыми индексами, задавать и учитывать их при приведении к каноническому виду. Оба вида абстрактных вычислений используют информацию о симметриях для приведения к каноническому виду и упрощения тензорных выражений.

Компонентные индексы требуют по сути скалярной системы компьютерной алгебры, возможно наличия простейших операций с матрицами. Фактически задаётся конкретная система координат и метрика. Поскольку все операции производятся компонентами, теряется информация о тензоре как целостном объекте, об его симметриях. Поэтому все операции с симметриями и приведение к каноническому виду должны быть выполнены на предыдущем этапе исследования.

В. Нотация

Использование систем компьютерной алгебры зачастую предполагает интерактивную работу пользователя. В этом случае удобство нотации играет важнейшую роль. Следует заметить, что исторически математическая запись тензоров следует нотации системы Т_РX, а именно тензор T_b^a записывается как $T^{\{a\}}_{\{b\}}$. Поэтому использование такой нотации было бы вполне естественным. Такой подход реализован

² Круглые скобки в (9) обозначают симметризацию.

³ Точка с запятой в (10) означает ковариантную производную.

в системе *Cadabra*. Однако *Cadabra* — специализированная система для тензорных вычислений. При введении тензорной нотации в системах компьютерной алгебры общего назначения следует учитывать ограничения этих систем (например, знак \wedge обычно зарезервирован и используется для возведения в степень).

Поскольку системы общего назначения работают с функциями, а основной внутренней структурой данных является список, то используется функционально-списочная нотация. В качестве имени функции может задаваться имя тензора, а ковариантные и контравариантные индексы задаются либо префиксом (например, как в *xAct*):

$T(a, -b)$,

либо позиционно (например, как в *Maxima*):

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

Возможно также использование ассоциативных списков, например таким образом:

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

Рассмотрим наиболее интересные для практического применения реализации тензорных вычислений в разных системах компьютерной алгебры.

C. Cadabra

Cadabra [<http://cadabra.phi-sci.com/>] относится к типу специализированных систем компьютерной алгебры. Область её специализации — теория поля. Поскольку сложные тензорные расчёты являются неотъемлемой частью теории поля, неудивительно, что поддержка тензорных расчётов находится в этой системе на высоком уровне.

Однако в теории поля оперируют в основном абстрактными индексами, компонентным вычислениям уделяется гораздо меньше внимания. Именно поэтому, наверное, компонентные вычисления пока ещё не реализованы в *Cadabra*. Хотя данная возможность и стоит в плане на реализацию.

Впрочем, компонентные вычисления требуют от системы компьютерной алгебры функционала системы общего назначения, который отсутствует в системе *Cadabra*.

D. Maxima

Maxima [<http://maxima.sourceforge.net/>] — одна из основных свободных систем компьютерной алгебры общего назначения. *Maxima* произошла от системы *Macsyma*, разрабатывавшейся в MIT с 1968 по 1982 годы.

В *Maxima* реализованы все три типа тензорных вычислений [11]:

- пакет *atensor* — безындексные алгебраические вычисления (в него заложен набор основных алгебр; основное предназначение — упрощение тензорных выражений при помощи манипуляций как с одноэлементными, так и с многоэлементными симметриями);
- пакет *ctensor* — компонентные вычисления (есть возможность манипулирования с метрикой, связностями; в пакет включён набор наиболее употребляемых метрик);
- пакет *itensor* — вычисления с использованием (абстрактных) индексов.

Реализация дублирует функциональность пакета *Macsyma* и фактически не развивается. Возможности данных пакетов вряд ли можно считать удовлетворительными на сегодняшний день.

E. Reduce

Reduce [<http://www.reduce-algebra.com/>], [12] — одна из старейших (из ныне живых) систем компьютерной алгебры общего назначения. В 2009 году сменила лицензию с коммерческой на лицензию BSD-типа. Правда, заметим, сделано это было достаточно поздно, сообщество в это время занималось другими свободными системами компьютерной алгебры, и поэтому большого преимущества от перехода к свободной лицензии система не получила.

В основную систему входят:

- пакет *atensor*, упомянутый ранее;
- пакет *redten* [<http://www.scar.utoronto.ca/~harper/redten.html>] предназначен для компонентных вычислений.

F. Maple

Система *Maple* [<http://www.maplesoft.com/products/Maple/index.aspx>] является коммерческой системой компьютерной алгебры общего назначения. Имеет в своём составе также и средства для численных вычислений.

- Система *Maple* имеет встроенные средства для манипуляции с тензорными компонентами, которые не уступают аналогичным средствам в других системах компьютерной алгебры. Фактически это два пакета:
 - пакет *tensor* изначально был направлен на решение задач общей теории относительности и предназначен для компонентных вычислений;

— в рамках мощного пакета *DifferentialGeometry* (вошедшего на данный момент в состав основной системы) присутствует подпакет *Tensor*, предназначенный также для компонентных вычислений. Его большим преимуществом является возможность использования не только тензорного анализа, но и всей мощи дифференциальной геометрии (например, использование симметрий групп и алгебр Ли);

- пакет *GRTensor II* [<http://grtensor.phy.queensu.ca/>] (лицензия GPL) является одним из самых мощных пакетов компонентных тензорных вычислений.

G. Mathematica

Mathematica [<http://www.wolfram.com/mathematica/>] — коммерческая система компьютерной алгебры общего назначения компании Wolfram Research. Содержит в своём составе целый комплекс расчётных и интерактивных инструментов (для создания математических учебных пособий).

- Пакет *MathTensor* [<http://smc.vnet.net/MathTensor.html>], [13] — коммерческий пакет, предназначенный в первую очередь для алгебраических манипуляций с тензорами.
- Пакет *Cartan (Tensors in Physics)* [<http://www.adinfinitum.no/cartan/>] — коммерческий пакет, предназначенный в первую очередь для вычислений в общей теории относительности. Поскольку выполняются вычисления в конкретных метриках, пакет оперирует компонентными индексами.
- Пакет *Ricci* [<http://www.math.washington.edu/~lee/Ricci/>] в основном поддерживает алгебраические манипуляции, также имеет и некоторые элементы компонентных операций. Находится в состоянии стагнации.
- Набор пакетов *xAct* [<http://www.xact.es/>] (лицензия GPL) охватывает операции как с абстрактными, так и с компонентными индексами.

В следующем разделе рассмотрим подробнее две системы компьютерной алгебры: *Cadabra* и *Maxima*.

Важнейшим критерием выбора этих двух систем стала лицензия — рассматривались только свободные системы компьютерной алгебры. Таким образом были исключены из рассмотрения пакеты расширения к коммерческим системам компьютерной алгебры (вне зависимости от лицензии на сами пакеты).

Наиболее развитая на данный момент свободная система компьютерной алгебры для тензорных манипуляций — *Cadabra*. Однако пока она не поддерживает

компонентные вычисления. Поэтому в качестве компаньона к ней выбрана система *Maxima*.

Возможным претендентом на рассмотрение могла стать система *Axiom*, однако на данный момент она распалась на несколько не вполне совместимых ответвлений.

IV. ОПЕРАЦИИ НАД ТЕНЗОРАМИ В СИСТЕМЕ CADABRA

В качестве демонстрации возможностей *Cadabra* рассмотрим выполнение разных операций над тензорами в этой системе. Напомним, что в текущей версии *Cadabra* отсутствует возможность работы с компонентами, поэтому компонентные операции рассмотрены не будут.

A. Безындексные вычисления

Зададим правила коммутации и проверим, что они выполняются:

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

- Случай коммутирующих тензоров:

B A;

1 := BA;

@prodsort! (%);

1 := AB;

- Случай антикоммутирующих тензоров:

D C;

2 := DC;

@prodsort! (%);

2 := -CD;

B. Голономные координаты

Покажем, что в случае согласованных метрики и связности ковариантная производная от метрического тензора равна нулю (см. (1)).

Зададим набор индексов, метрику и частную производную:

```
{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.
```

Запишем ковариантную производную через символы Кристоффеля и символы Кристоффеля через метрический тензор, чем собственно и зададим согласование метрики со связностью:

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

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

```
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}\} );
```

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

Подставим выражение символа Кристоффеля через метрический тензор в выражение для ковариантной производной:

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

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

и раскроем скобки:

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

$$\nabla := \partial_c g_{ab} - \frac{1}{2} g_{ad} g^{de} \partial_b g_{ec} - \frac{1}{2} g_{ad} g^{de} \partial_c g_{be} + \frac{1}{2} g_{ad} g^{de} \partial_e g_{bc} - \frac{1}{2} g_{db} g^{de} \partial_a g_{ec} - \frac{1}{2} g_{db} g^{de} \partial_c g_{ae} + \frac{1}{2} g_{db} g^{de} \partial_e g_{ac};$$

Подымаем и опускаем индексы до тех пор, пока не убедим все метрические тензоры, о чём говорит двойной восклицательный знак:

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

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

Далее приведём выражение к каноническому виду и приведём подобные. В результате получим ноль, как и ожидалось:

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

$$\nabla := 0;$$

С. γ -матрицы

Cadabra имеет развитые средства для работы с γ -матрицами любой размерности. Для определённости будем использовать γ -матрицы дираковских 4-спиноров.

Для упрощения вычислений зададим набор действий, выполняющихся после каждой операции:

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

Зададим индексы и пробегаемые ими значения:

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

Размерность пространства будет использоваться при нахождении следа символа Кронекера.

При задании γ -матриц указывается метрика (см. (2)):

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

Теперь продемонстрируем несколько симметричных тождеств, которым удовлетворяют γ -матрицы.

- Уравнение Клиффорда–Дирака (2):

$$\gamma_a \gamma_b + \gamma_b \gamma_a;$$

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

Алгоритм @join преобразует попарные произведения γ -матриц в сумму γ -матриц высших валентностей (см. (3)). Дополнительный аргумент expand указывает на то, что учитываются правила антисимметризации для γ -матриц:

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

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

- Свёртка двух γ -матриц (4):

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

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

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

$$2 := 4;$$

- Тожество (5):

```
\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! (%) ;
```

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

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

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

- Тожество (6):

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

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

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

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

- Тожество (7):

```
\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};
```

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

- Более сложное тождество:

$$\gamma_{ab} \gamma_{bc} \gamma_{de} \gamma_{ea} = -4\gamma_{cd} + 21g_{cd}\hat{I}$$

в Cadabra будет иметь вид:

```
\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};
```

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

D. Одноэлементные симметрии

В качестве примера одноэлементной симметрии рассмотрим симметрии тензора Римана (8). Для этого зададим вначале симметричные свойства с помощью диаграммы Юнга:

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

В данном примере симметрия имеет вид $\begin{array}{|c|c|} \hline a & c \\ \hline b & d \\ \hline \end{array}$.

Далее произведём симметричную и антисимметричную перестановку индексов. Алгоритм `@canonicalise` приводит операнд к канонической форме, учитывая одноэлементные симметрии.

```
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! (%) ;
```

$$2 := 0;$$

Е. Многоэлементные симметрии

Работу с многоэлементными симметриями продемонстрируем на примере тензора Римана.

Введём обозначения для индексов и производной:

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

Симметрию можно задавать с помощью диаграммы Юнга $\begin{array}{|c|c|c|} \hline a & c & e \\ \hline b & d & \\ \hline \end{array}$, как в предыдущем случае:

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

Однако удобнее использовать следующую нотацию:

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

Аналогично поступим и с ковариантной производной от тензора Римана, удовлетворяющей дифференциальному тождеству Бьянки:

```
\nabla_{e}{R_{a b c d}}::SatisfiesBianchi.
```

Проверим первое тождество Бьянки (9).

```
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(%) {ModuloMonoterm}:
```

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

$$1 := 0;$$

Теперь продемонстрируем выполнение второго (дифференциального) тождества Бьянки (10):

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

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

```
@young_project_tensor!2(%) {ModuloMonoterm}:
```

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

$$2 := 0;$$

V. ПРИМЕР ТЕНЗОРНЫХ ВЫЧИСЛЕНИЙ В МАХИМА

Поскольку Cadabra на данный момент не поддерживает компонентные вычисления, продемонстрируем их в системе Maxima. В качестве примера рассмотрим запись уравнений Максвелла в цилиндрических координатах в голономном базисе [1].

Загрузим вначале написанный нами небольшой пакет, содержащий определения для дифференциальных операторов:

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

Зададим цилиндрическую систему координат:

```
(%i2) ct_coordsys(polar cylindrical)$
```

Посмотрим компоненты метрического тензора g_{ij} :

```
(%i3) lg;
```

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

```

Вычислим и посмотрим компоненты метрического тензора g^{ij} :

```
(%i4) cmetric()$
```

```
(%i5) ug;
```

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

```

Зададим необходимые векторы и определим их зависимость от координат. Из-за ограничений Maxima обозначим j^1 через j1, а j_1 через 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))$
```

Теперь вычислим все части уравнений Максвелла в цилиндрических координатах:

```
(%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) 
$$\begin{pmatrix} \frac{\frac{d}{d\theta} H}{|r|} - \frac{3 - \frac{d}{dz} H}{|r|} \frac{2}{c} + \frac{\frac{d}{dt} D1}{c} - \frac{4\pi j1}{c} \\ -\frac{\frac{d}{dr} H}{|r|} - \frac{3 - \frac{d}{dz} H}{|r|} \frac{1}{c} + \frac{\frac{d}{dt} D2}{c} - \frac{4\pi j2}{c} \\ \frac{\frac{d}{dr} H}{|r|} - \frac{2 - \frac{d}{d\theta} H}{|r|} \frac{1}{c} + \frac{\frac{d}{dt} D3}{c} - \frac{4\pi j3}{c} \end{pmatrix}$$

```

```
(%i19) Rot(E)
```

```
+ diff(transpose(matrix(B)), t)/c;
```

$$(\%o19) \left(\frac{\frac{d}{dt} E}{c} \frac{3 - \frac{d}{dz} E - 2}{|r|} + \frac{\frac{d}{dt} B1}{c} \right) \\ \left(\frac{\frac{d}{dt} B2}{c} - \frac{\frac{d}{dr} E}{|r|} \frac{3 - \frac{d}{dz} E - 1}{|r|} \right) \\ \left(\frac{\frac{d}{dr} E}{|r|} \frac{2 - \frac{d}{dt} E - 1}{|r|} + \frac{\frac{d}{dt} B3}{c} \right)$$

Таким образом, результат совпал с полученными в [1] аналитическими выражениями.

VI. ЗАКЛЮЧЕНИЕ

Статья вызвана к жизни тем, что авторы хотели провести громоздкие тензорные вычисления в системе компьютерной алгебры. Поиск соответствующей системы заставил сформулировать набор критериев, которым должна удовлетворять подобная система компьютерной алгебры.

Из-за наличия нескольких типов тензорных расчётов полная реализация тензорных вычислений в системах компьютерной алгебры требует широкого набора возможностей. К сожалению, на данный момент фактически отсутствуют системы с полностью удовлетво-

рительной поддержкой тензоров.

Компонентные тензорные вычисления практически не требуют дополнительных возможностей от универсальной системы компьютерной алгебры. Поэтому пакеты, реализующие данный функционал, представлены наиболее широко (например, *Maxima*, *Maple*, *Mathematica*).

Тензорная нотация сильно отличается от привычной функциональной нотации, которую использует подавляющее большинство систем компьютерной алгебры. Поэтому для эффективной работы с тензорами представляется необходимым иметь специализированную систему (либо специализированную надстройку над универсальной системой), поддерживающую естественную тензорную нотацию (например, *Cadabra*).

В результате не удалось найти систему, полностью удовлетворяющую потребностям тензорного исчисления. Авторы пока остановились на наборе из специализированной системы *Cadabra* и универсальной системы компьютерной алгебры (на данный момент *Maxima*).

-
- [1] Kulyabov D. S., Korolkova A. V., Korolkov V. I. Maxwell's Equations in Arbitrary Coordinate System // Bulletin of PFUR. Series "Mathematics. Information Sciences. Physics". — 2012. — no. 1. — P. 96–106.
- [2] Кулябов Д. С., Немчианова Н. А. Уравнения Максвелла в криволинейных координатах // Вестник РУДН. Серия «Математика. Информатика. Физика». — 2011. — № 2. — С. 172–179.
- [3] Сарданашвили Г. А. Современные методы теории поля. Т. 2. Геометрия и классическая механика. — М.: УРСС, 1998.
- [4] Cartan E. The Theory of Spinors. — Paris: Hermann, 1966.
- [5] Гердт В. П., Тарасов О. В., Ширков Д. В. Аналитические вычисления на ЭВМ в приложении к физике и математике // Успехи физических наук. — 1980. — Т. 130, № 1. — С. 113–147. — URL: <http://ufn.ru/ru/articles/1980/1/d/>.
- [6] Барут А., Рончка Р. Теория представлений групп и её приложения. — М.: Мир, 1980.
- [7] Пенроуз Р., Риндлер В. Спиноры и пространство-время. Два-спинорное исчисление и релятивистские поля. — М.: Мир, 1987. — Т. 1. — 528 с.
- [8] Computer Algebra Information Network. — URL: <http://www.computeralgebra.nl/systemoverview/special/systems.html>.
- [9] Кулябов Д. С., Кокотчикова М. Г. Аналитический обзор систем символьных вычислений // Вестник РУДН. Серия «Математика. Информатика. Физика». — 2007. — № 1–2. — С. 38–45.
- [10] Sevastianov L. A., Kulyabov D. S., Kokotchkova M. G. An Application of Computer Algebra System Cadabra to Scientific Problems of Physics // Письма в ЭЧАЯ. — 2009. — Т. 6, № 7 (156). — С. 39–44.
- [11] Toth V. Tensor Manipulation in GPL Maxima. — 2005. — arXiv : cs/0503073v2.
- [12] Gerdt V. P., Tiller P. A Reduce program for symbolic computation of Puiseux expansions. — Dubna : Joint Inst. Nuclear Res., 1991.
- [13] Parker L., Christensen S. M. MathTensor: a system for doing tensor analysis by computer. — Addison-Wesley, 1994. — 379 p.