ALGEBRAIC FIELD THEORY

A MATHEMATICAL CONSTRUCTION OF THE VACUUM

ROBERT HANS IHDE†
MUNICH (GERMANY), AUGUST 2018

ABSTRACT

All algebras of quantum theory are algebras over a field or briefly K-algebras of a binary operation, which are defined as constant invariants over the Poincaré group. The Christoffel symbols occurring in the classical geodetic equation can be understood as a representation of a K-algebra. In contrary to the constant algebras of quantum theory, the K-algebra of the Christoffel symbols is a function of space-time. A mathematical, conformal unification of geometry and algebra requires a corresponding dependence on space-time for the quantum algebras. Assuming the existence of an algebraic field, based on a changing binary operation with a feedback to the domain, a quantum mechanical vacuum equation for gravity is established. The vacuum equation structurally follows a generalized, pseudo-linear Dirac-Maxwell system with additional algebraic constraints. A physical existence of the algebraic field, as the counterpart to the geometric field of gravity, can be falsified by the experiment. The part of the spin in the magnetic moment of a particle would then depend on its acceleration, since the algebraic field should influence the spin algebra accordingly.

1. INTRODUCTION

"How can it be, that mathematics, being after all product of human thought which is independent of experience, is so admirable appropriate to the objects of reality?"

Konrad Zuse[1] had a simple information-theoretical thesis (IT) as an answer to Albert Einstein's[2] epistemological question. Nature could already function like an algebraic computer on the smallest level. However, Zuse's approach of a classical computer does not take quantum mechanical principles[3] into account and the connection to gravity also remained open. The Feynman/Stückelberg Y-graph[4] of the interaction of two elementary particles, e.g. the annihilation of an electron with a positron to a photon, is understood in the IT picture as a finger pointing of nature to a deeper, binary operation which then takes place on the Planck length ($\approx 10^{-33}$ cm). At an energy of about 14 Tev available today, any interaction that can be resolved, would still be a package of several quadrillions of such smallest operations.

The world of a modern, almost natural, computer game on the monitor and what is going on parallel on the processor and main memory seem to be two completely separate worlds, but one requires the other. A similar situation occurs when you want to connect the current elementary particle physics with a speculative IT physics on the Planck level. Therefore, analogies and new points of view are conveyed simultaneously with mathematics in order to facilitate the introduction to algebraic thinking in relation to physics. This is not absolutely necessary, but was helpful as a model for the development of this theory. Anyway, only the mathematics presented here is of importance and the experiment is decisive, independent of the concrete picture.

Are the terms 'reality' and 'virtuality' based on the same mathematical principles and can this be proven experimentally?

According to Karl Popper[5], a physical theory cannot be proved in principle, but only falsified. This work should therefore continue the path inspired by Konrad Zuse, towards a falsifiable algebraic field theory.

The approach presented here is far removed from all current main directions of physics. Rather, it should be understood as a restriction from an algebraic point of view, similar to higher programming languages that lead to the machine language.

---

† roberthans.ihde@gmail.com, preprint, draft, subject to changes and error corrections

To realize this theory, an axiomatic, bottom-up approach is chosen. Only a generalized binary operation is assumed, mediated by arbitrary K-algebras, which can change themselves depending on the domain. Everything else is derived from these few axioms.

Binary operations can be generalized by K-algebras, which can be constructed by a generating base. In quantum theory, K-algebras contribute substantially to their properties, for example the γ matrices as generators in the Dirac equation[6] for the electron, the Pauli σ matrices[7] for the spin, or the Gell-Mann's λ matrices[8], describing the quark color symmetries. There are further structural clues in physics which support an algebraic view. The Maxwell equations[9] can be derived as generalized Cauchy-Riemann equations from a subspace of Biquaternions as generators[10]. The Dirac equation itself can also be understood as the Cauchy-Riemann system of the γ matrices[11]. In physics such systems are therefore also called 'Dirac-Maxwell' systems. Solutions of general relativity[12] have the property of conserving angles under physical transformations, a genuine property of conformal Cauchy-Riemann systems. These mathematical facts already point in the direction that the formula structures of quantum mechanics and gravity can have their origin in algebraic constraints of a generalized, binary operation determined by K-algebras. For this purpose, the K-algebra itself must be able to change in the form of a local algebraic field with a feedback to the domain. The complementary image is that of a simple programmable transistor (or artificial neuron) with two inputs and one output. The output is determined by the two inputs but also by the changing weights of the K-algebra, which themselves depend on the domain. It will be shown in the following that such a construct follows a pseudo-linear Cauchy-Riemann system (math.) or generalized Dirac-Maxwell system (phys.) for quantum gravity.

In order to clarify the overall picture, this will first be demonstrated using the example of complex numbers. A complex number can also be described as a pseudo-scalar product between the vector of the generators (e, i) and a vector of the real field numbers, where the operator T symbolizes the transposition of a matrix or vector. In the literature, the complex one with the real one is often identified on the basis of the same algebraic properties (e ≡ 1). Via the strict mathematical view, all generators are always of a different kind than the real numbers.

$$2\,\mathrm{e} + 3\,\mathrm{i} \;=\; \begin{pmatrix} \mathrm{e} \\ \mathrm{i} \end{pmatrix}^{T} \cdot \begin{pmatrix} 2 \\ 3 \end{pmatrix} \tag{1.1}$$

Thus, this concept is also suitable for the mathematical description of a memory (the generators here represent two cells) with the respective internal states (the real numbers).

The K-algebra $\mathbb{C}$ of the complex numbers can be represented as a binary operation using the vector of the generating base and the Kronecker/Zehfuss-product (symbol $\otimes$) as follows (classical construction of a K-algebra from the generating base elements by embedding into the domain). Thereby the generators get their actual meaning. Remark: In the quantum mechanical view two states (left side) are generated from one state (right side) by means of the rectangular matrix as ascending operator (in the Bra-Ket notation: $|\beta, \beta\rangle = \boldsymbol{A}|\beta\rangle$). The algebraic field can therefore be interpreted also as a creation or annihilation operator field.

$$\begin{pmatrix} e \\ \mathrm{i} \end{pmatrix} \otimes \begin{pmatrix} e \\ \mathrm{i} \end{pmatrix} \;\coloneqq\; \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ -1 & 0 \end{pmatrix} \cdot \begin{pmatrix} e \\ \mathrm{i} \end{pmatrix} \tag{1.2}$$

All eight weights of the rectangular matrix unambiguously determine the K-algebra of the complex numbers $\mathbb{C}$. Changing these weights results in different number systems, i.e. a different K-algebra or as function of the domain as a local algebraic field.

The complex multiplication can be determined freely from the generators, by combining the rectangular matrix with the Kronecker product, only by means of the vectors of the field numbers (this follows already from the definitions 1.1 and 1.2).

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ -1 & 0 \end{pmatrix}^{T} \cdot (p \otimes q) = r \qquad p, q, r \in \mathbb{R}^2 \tag{1.3}$$

The geodetic equation follows the same structure when the Christoffel symbols are mapped into a corresponding rectangular 16x4 matrix $\boldsymbol{\Gamma}$ and then brought into the form of a matrix vector product. The connection $\boldsymbol{\Gamma}$ can then be interpreted as a local space-time dependent K-algebra. The point above the location vector q denotes the derivation according to the eigentime.

$$-\boldsymbol{\Gamma}^{T}(\dot{q} \otimes \dot{q}) = \ddot{q} \tag{1.4}$$

Another heuristic indication of a local dependency between algebra and metric are the Diracian γ matrices as generators. These are resulting as division algebra from the linearized Klein-Gordon equation from the Minkowski metric. The following constraint[6] must be fulfilled with the help of the anti-commutator brackets:

$$\tfrac{1}{2}\{\gamma^{j},\ \gamma^{k}\} = G^{jk}E \tag{1.5}$$

Those peculiarities are now the mathematical motivation to infer a more general physical relation between any K-algebra as local algebraic field and the related metric. While Dirac assumed a definite, constant metric in order to deduce the necessary algebra, the opposite way is taken here. It is investigated under which mathematical conditions a local algebraic field induces another local metric field as geometric conservation quantity[13].

2. DEFINITIONS

For the formulation of the Algebraic Field Theory no new mathematics is necessary, but the use of different mathematical tools, which are not very common in physics at present. This concerns the use of the rectangular matrix calculus in interaction with the Kronecker product as well as the use of shift index operators, hereinafter referred to as `RKS notation`. The RKS notation is optimized for programming and at the same time reduces the amount of formula writing for the Algebraic Field Theory. The following additional notations are used. Standard lower-case letters like `x` are variables from the field numbers, italic lower case standard letters like `$q$` are vectors. Capitalized letters in italics like `$M$` should denote square matrices, bold letters in italics and capitalized Greek letters like `$\boldsymbol{A}$` or `$\boldsymbol{\Gamma}$` should represent rectangular matrices. The Greek little `$\beta$` is reserved for the vector of the generating base to symbolize that this vector is of a different kind than the vectors defined over the field. The Einstein sum convention common in physics as well as the symbols for differential operators are used, as far as no ambiguities occur. Furthermore, the Euclidean unit vectors are called `$e$` and the quadratic unit matrix is called `$E$`. In case of transition to affine manifolds these are to be generalized as location-dependent Vielbein, by replacing $e_j \rightarrow e_j(q)$, i.e. all operators are generally dependent on the domain. Exceptions to this notation are known entities such as Dirac's Gamma Matrices `γ`.

Definition for a vector over the field

$$q := q^j e_j \qquad (j, n \in \mathbb{N},\ q^j \in \mathbb{K},\ e_j \in \mathbb{K}^n,\ e^j e_k = \delta^j_k) \tag{2.1}$$

Definition for the vector of the generators ('atomic memory cells')

$$\beta := \beta^j e_j \qquad (\beta^j \notin \mathbb{K} \wedge d\beta^j = \partial\beta^j := 0 \in \mathbb{K}) \tag{2.2}$$

Note: This definition results from the generalization of the symbol `i` for the complex numbers. The infinitesimal difference of `i` vanishes, because this symbol is an atomic entity. In the IT picture the β^j are seen as atomic memory cells, which themselves do not belong to space-time and do not need any further topological assumptions. Thus, the metric operator for up and down indexing is only possible for (2.1). For (2.2) an own conjugation operator is necessary, which is derived later directly from the K-algebra, if certain conditions are fulfilled. The metric space-time is generated virtually and exclusively by the algebraic relations between the atomic memory cells, comparable to a computer game. The individual cells must not be equated with a quantized space-time. These are to be regarded as degrees of freedom. The postulates for quantization are not assumed, but are identified in the following as a specific variant from the class of all algebraic field theories.

Definition of a generalized complex number ('memory state')

$$\tilde{q} := \beta^T q = q^T \beta \tag{2.3}$$

Definition of the mapping of variables with triple indices to a rectangular matrix

$$\boldsymbol{A} := \left(e_j \otimes e_k \otimes e^l\right) a^{jk}_l \qquad a^{jk}_l \in \mathbb{K} \tag{2.4}$$

For clarification an example of any two-dimensional K-algebra:

$$\boldsymbol{A} = \begin{pmatrix} a_1^{11} & a_2^{11} \\ a_1^{12} & a_2^{12} \\ a_1^{21} & a_2^{21} \\ a_1^{22} & a_2^{22} \end{pmatrix} \quad \text{e.g. for the complex numbers } \mathbb{C} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ -1 & 0 \end{pmatrix}$$

The use of standard complex numbers already as field numbers, as usual in physics, may have disadvantages in treatment or in explicit computer calculations with algebraic operators. The appendix (A.1) describes a procedure that always maps Cartesian products of K-algebras (f.e. Biquaternions) to a corresponding, higher dimensional algebra over the real numbers.

In addition to the superscript operator ′*T*′ of the transposition, three more superscript operators are needed for the RKS notation.

The operator ′*I*′ should represent the Moore-Penrose Inverse[14] of a rectangular matrix, which is defined by the following pseudo-inverse property. In a normal square matrix, the same symbol means the usual matrix inverse.

$$\boldsymbol{A}^I = \boldsymbol{A}^I \boldsymbol{A}\, \boldsymbol{A}^I \quad \wedge \quad \boldsymbol{A} = \boldsymbol{A}\, \boldsymbol{A}^I \boldsymbol{A} \tag{2.5}$$

From the six possible permutations of the trivalent indices (including the identity), two index shift operators can already be selected as the base. The first operator exchanges the first two indices bilaterally against each other from the left, the second operator shifts each index one position to the right. Note, that the classic index shifts (in brackets) are only valid for Euclidean metrics.

Definition Superscript Shift Operator B ("bilateral shift")

$$\boldsymbol{A}^B := \left(e_j^T \otimes E \otimes e_{\mathrm{j}}\right) \boldsymbol{A} \qquad (a_{\mathrm{l}}^{\mathrm{jk}})^B = a_{\mathrm{l}}^{\mathrm{kj}} \tag{2.6}$$

Definition Superscript Shift Operator R ("right shift, rotate thru carry")

$$\boldsymbol{A}^R := \left(e_{\mathrm{j}} \otimes E\right) \boldsymbol{A}^T \left(E \otimes e_{\mathrm{j}}\right) \qquad (a_{\mathrm{l}}^{\mathrm{jk}})^R = a_{\mathrm{k}}^{\mathrm{lj}} \tag{2.7}$$

When executing the Superscript operators sequentially, it must be payed attention to the position and to their rules of permutation (A.2). The order of execution must always be read from left to right.

$$\boldsymbol{A}^{BRI} := ((\boldsymbol{A}^B)^R)^I \tag{2.8}$$

The definitions of the shift operators already imply relations (A.2) and in interaction with the Kronecker product further identities can be derived (A.3).

The Neudecker vector function[15] for transforming any matrix into a vector, arranged by the columns, is defined as follows.

$$vec(\boldsymbol{A}) := \sum_{\mathrm{j}=1}^{\mathrm{n}_c} \left(e_{\mathrm{j}}(\mathrm{n}_c) \otimes E(\mathrm{n}_r \times \mathrm{n}_r)\right) \boldsymbol{A}(\mathrm{n}_r \times \mathrm{n}_c)\ e_{\mathrm{j}}(\mathrm{n}_c) \tag{2.9}$$

A general construction of a K-algebra from a generating base can be guaranteed with the following definition. The sub index at the Kronecker symbol stands as a parameter for the K-algebra, with which the Kronecker product on the left side is to be replaced with the respective context of a corresponding rectangular matrix on the right side.

In the following, the abbreviated term 'algebra' or 'algebraic field' is therefore used as a synonym for the representation of a domain-dependent K-algebra by means of a rectangular matrix (see 1.2 for the complex numbers as special case).

$$\beta \otimes_{\mathbb{A}} \beta := \boldsymbol{A}\,\beta \tag{2.10}$$

An affine connection can now be defined according to the same scheme, using the Levi-Civita Operator ∇ and the metric tensor G (A.4). Here the generating base vector is used as a local frame to define the Christoffel symbols as a rectangular matrix $\boldsymbol{\Gamma}$.

$$(G\,\nabla) \otimes \beta := \boldsymbol{\Gamma}\,\beta \tag{2.11}$$

This allows the definition of a derivative, minimally coupled with $\boldsymbol{\Gamma}$, which is lifted into the rectangular matrix form in order to arrive later at a generator free representation (it follows $\boldsymbol{\nabla}\beta = 0$).

$$\boldsymbol{\nabla} := (G\,\nabla) \otimes E - \boldsymbol{\Gamma} \tag{2.12}$$

The model of a computer with only one memory forces the identification of the dual space with the outgoing space. Anti-particles and particles are in the same space-time. Complex numbers follow the 'one memory' principle, because the dual space of the complex numbers is identical to that of the complex numbers. The dual base of the generators is formulated in the Algebraic Field Theory with the help of the pseudo-inverse base. Due to the identity of the dual space with the original space, the pseudo-inverse basis can also be expressed by means of the original basis.

The corresponding transformation matrix is then the "Hermitian" operator $H_{\mathbb{A}}$ (or "conjugation") for the related K-algebra $\mathbb{A}$ and is generalized according to,

$$\begin{pmatrix} e \\ i \end{pmatrix}^I = \tfrac{1}{2}\,(e \quad -i) = \begin{pmatrix} e \\ i \end{pmatrix}^T \cdot \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \cdot \tfrac{1}{2} \quad \text{(complex numbers)} \tag{2.13}$$

$$\beta^I := \beta^T \cdot H_{\mathbb{A}} \cdot \mathrm{m}_{\mathbb{A}}^{-2} \quad (K_{\mathbb{A}}, \mathrm{m}_{\mathbb{A}} \in \mathbb{K})$$

where $\mathrm{m}_{\mathbb{A}}$ is an algebra-induced gauge coefficient for the dual base. Chapter 5 shows later that the existence of such a generalized dual operator is subject to additional constraints.

For the gauge coefficient of the domain base the pseudo-Riemannian metric is assumed, thus structurally equal to the algebraic base (2.13), if the field identity $1_{\mathbb{K}}$ is added, analogous to the related identity (algebraic one) $\tilde{o}_{\mathbb{A}}$ of the related K-algebra. The metric tensor G is thus interpreted as the "conjugation" operator of the domain:

$$\mathrm{m}_{\mathbb{K}}^2(q)\,1_{\mathbb{K}} = q^T\,G_{\mathbb{K}}\;\;q \quad \wedge \quad \mathrm{m}_{\mathbb{A}}^2(\beta)\,\tilde{o}_{\mathbb{A}} = \beta^T\;H_{\mathbb{A}}\;\beta \tag{2.14}$$

If a dual algebraic base is available, a generalized, inverse complex number can be defined analogous to the complex numbers as follows:

$$\tilde{q}^I := \beta^I q\;\mathrm{m}_{\mathbb{K}}^{-2}(q) \tag{2.15}$$

3. GENERAL

Each K-algebra of a binary operation constructed by the definitions (2.3) and (2.10) can be represented as a combination of a rectangular matrix vector product and the Kronecker product.

$$\tilde{p} \otimes_{\mathbb{A}} \tilde{q} = (\beta^T p) \otimes_{\mathbb{A}} (\beta^T q) = (\beta^T \otimes_{\mathbb{A}} \beta^T)(p \otimes q) = \beta^T \boldsymbol{A}^T (p \otimes q) \tag{3.1}$$

The binary operation can also be reversed in the order in which the bilaterally shifted algebra is performed.

$$\tilde{q} \otimes_{\mathbb{A}} \tilde{p} = \tilde{p} \otimes_{\mathbb{A}^B} \tilde{q} \tag{3.2}$$

It follows that a K-algebra is commutative, if the rule applies:

$$\boldsymbol{A}^B = \boldsymbol{A} \tag{3.3}$$

Solutions for algebraic problems therefore usually have a left- and right-handed version, if the K-algebra is not commutative. In the following, only a one-handed solution is given. The other one can be determined by replacing the rectangular matrix $\boldsymbol{A} \longmapsto \boldsymbol{A}^B$ in the solution.

Using the *vec* function and the Jacobi matrix representation of the Levi-Civita operator, the relationship (2.11) can be symbolically rewritten as follows:

$$vec\left(\frac{d\beta}{dq}\right) = vec(\nabla^T \otimes \beta) = \nabla \otimes \beta = (G^I \otimes E)\ \boldsymbol{\Gamma}\,\beta \tag{3.4}$$

This allows the infinitesimal, total Levi-Civita difference of β to be determined, using Kronecker/shift operator symmetries (A.2/A.3):

$$d\beta = \boldsymbol{\Gamma}^{RRT}(\beta \otimes G^{IT})\, dq = \boldsymbol{\Gamma}^{RRT}(E \otimes G^{IT} dq)\,\beta := dQ\,\beta \tag{3.5}$$

This has an effect on (2.10). Looking at this infinitesimal change of the expression, the new base, infinitesimally shifted by dq around $d\beta$, must apply:

$$(\beta + d\beta) \otimes_{\mathbb{A}+d\mathbb{A}} (\beta + d\beta) = (\boldsymbol{A} + d\boldsymbol{A})(\beta + d\beta) \tag{3.6}$$

The terms of the second order fall away on both sides, then follows:

$$(\boldsymbol{A} + d\boldsymbol{A})\beta + (E \otimes dQ + dQ \otimes E)\boldsymbol{A}\beta = (\boldsymbol{A} + d\boldsymbol{A})\beta + \boldsymbol{A}dQ\beta \tag{3.7}$$

In order for this to be fulfilled, it must hold:

$$\boldsymbol{A}dQ = (E \otimes dQ + dQ \otimes E)\,\boldsymbol{A} \tag{3.8}$$

For the total difference of the algebra together with (3.8) ensues:

$$d\boldsymbol{A} = \partial\boldsymbol{A} + (E \otimes dQ + dQ \otimes E)\boldsymbol{A} + \boldsymbol{A}dQ = \partial\boldsymbol{A} + 2\boldsymbol{A}dQ \tag{3.9}$$

The following also counts:

$$d(\beta \otimes_{\mathbb{A}} \beta) = d\boldsymbol{A}\,\beta + \boldsymbol{A}d\beta \tag{3.10}$$

From this follows by means of (3.8):

$$(E \otimes dQ + dQ \otimes E)\boldsymbol{A} = d\boldsymbol{A} + \boldsymbol{A}dQ = \boldsymbol{A}dQ \tag{3.11}$$

This can only be fulfilled, if it pertains to the algebra:

$$d\boldsymbol{A} = 0 \tag{3.12}$$

From this follows immediately from equation (3.9):

$$\partial \boldsymbol{A} = -2\boldsymbol{A}dQ = -2\,\boldsymbol{A}\,\boldsymbol{\Gamma}^{RRT}(E \otimes G^I)\,dq \tag{3.13}$$

In order for (3.6, 3.10) both to be fulfilled, the following must apply between $\boldsymbol{A}$ and $\boldsymbol{\Gamma}$:

$$½\frac{\partial \boldsymbol{A}}{\partial q} = -\boldsymbol{A}\,\boldsymbol{\Gamma}^{RRT}(E \otimes G^I) \tag{3.14}$$

The source of geometric space-time distortion is the energy impulse tensor, according to general relativity. In Algebraic Field Theory gravity is coupled with a changing algebraic field, similar to the coupling of a magnetic field to an electric field. The operator $\boldsymbol{\Gamma}$ must vanish according to (3.14) if the algebraic field is constant and non-zero or has an extremum.

The standard deduction of the operator $\boldsymbol{\Gamma}$ from the metric G for a holonomic base (here in the RKS notation, a factor G^I is here missing because drawn to (2.11) for easier calculations using the degree of freedom for the definition of $\boldsymbol{\Gamma}$):

$$\boldsymbol{\Gamma} = ½\left((\partial \otimes G)^R + (\partial \otimes G)^{RR} - (\partial \otimes G)\right) \tag{3.15}$$

The equation (3.14) can then be replaced on the right side by this relation and results in a non-linear differential equation of the first order for the algebraic field. If the metric tensor itself is a function of the algebraic field $G(\boldsymbol{A})$, describing the "self-interaction" of the vacuum state, then this differential equation depends only on the algebra (see chapter 6). This has to be considered as a constraint for the later vacuum solution. When interacting with an external metric field, the tensor G can alternatively be determined semi-classically via general relativity.

$$\frac{\partial \boldsymbol{A}}{\partial q} = -\boldsymbol{A}\left((\partial \otimes G)^T + (\partial \otimes G)^{RT} - (\partial \otimes G)^{RRT}\right)(E \otimes G^I) \tag{3.16}$$

The equation must then also be valid for all quantum algebras. These would have to change correspondingly under high accelerations or strong gravitation and thus it is possible to physically falsify this mathematically justified speculation of the existence of algebraic fields. Without knowing the explicit solutions, the effect can be estimated at least structurally.

For a particle, which moves with nearly speed of light on a circle with radius vector r, the geodetic equation applies in classical approximation:

$$-\boldsymbol{\Gamma}^T(\dot{q} \otimes \dot{q}) = \ddot{q} \approx \frac{c^2}{|r|^2}\,r = -\boldsymbol{\Gamma}^T(c \otimes c) \text{ and } c^T r = 0 \tag{3.17}$$

If the particle now passes through two different radii, the change of the geometric field should be in first approximation dependent on the amount of the difference between the two different radii, if the difference is small compared to the total radius.

$$|\delta\boldsymbol{\Gamma}| \approx \frac{|\delta r|}{|r|^2} \qquad \delta r \ll r \tag{3.18}$$

This relation used in equation (3.16) then gives approximately for the change of the algebra with an experimental factor n_{exp} to be determined:

$$\delta\boldsymbol{A} \approx \mathrm{n}_{exp}\,\boldsymbol{A}\,|\delta r|^2 \tag{3.19}$$

The magnetic moment u of an electron is the sum of the moment of the orbital angular momentum l and of the spin s and the Landé factor g_L (with e=charge, m=mass):

$$u = \frac{e}{2m}(l + g_L \cdot s) \tag{3.20}$$

In quantum theory, the difference of the magnetic moment between two different orbital radii depends only on the orbital angular momentum, since the spin does not depend on the radius or on the respective acceleration (centrifugal force):

$$\delta u_{QT} = \frac{e}{2m}\delta l \tag{3.21}$$

Algebraic Field Theory predicts an additional term resulting from the change in the spin K-algebra which is a possibility for falsification. From the point of view of current quantum theory, this would look like a change in the Landé factor as a function of the orbit radius:

$$\delta u_{AF} \approx \frac{e}{2m}\left(\delta l + g_L n_{exp}|\delta r|^2 s\right) \tag{3.22}$$

4. ALGEBRAIC IDENTITIES

In the first order the vacuum does not interact with other states. From the algebraic point of view this can only be a neutral element. For this there are two possibilities, the algebraic 'one' or the algebraic 'zero'. The latter can be excluded for the vacuum, since the zero-state extinguishes every other state. Within quantum theory, the zero-state is hardly mentioned, since trivially the probability there is exactly zero. The 'nothing' does not exist. In an Algebraic Field Theory of gravity, this is generally no longer excluded.

If no further additional assumptions are made, besides the 'global' zero, additional 'local' zero-states can theoretically exist with non-vanishing probability. At first this seems to be a physical abstrusity. It would mean that space-time could be destroyed. In quantum theory, particles can be created from vacuum and destroyed again, and thus this property would be an understandable consequence of quantum gravity being transferred to space-time itself. The continuing expansion of the universe indicates that space and time are continuously created. What can be created should therefore also be able to be destroyed. This also does not contradict the special as well as the general relativity theory, because both theories presuppose classically a stable and coherent space-time. The tunnel effect in the physics of elementary particles would then find its interpretation in the IT picture. If a small space-time range is erased by means of a zero-state field, then this could force the particle to jump because of the preservation of information. A universe with unstable space-time is automatically no longer deterministic, since all trajectories of the particles are no longer complete.

ALGEBRAIC ZERO-STATE

All K-algebras share the 'global' zero according to the definition (3), which is often identified with the zero of the field (in the strict sense both are different because of the different structure).

$$\forall \tilde{q}, \mathbb{A}: \quad \tilde{q} \otimes_{\mathbb{A}} \tilde{n} = \tilde{n} \otimes_{\mathbb{A}} \tilde{q} = \tilde{n} := 0 \tag{4.1}$$

The 'global' zero is theoretically the only shared state to all K-algebras. In the algebraic interpretation the physical equivalent is the beginning of the universe, because only in the beginning everything was equal and common. The probability here is exactly equal to zero according to definition and thus the vacuum cannot acquire this global zero-state and must always have a different state.

Nevertheless, the global zero is suitable for Algebraic Field Theory as the origin of an excellent reference system, the view from the same beginning, so to speak.

The algebraic constraints for other zeros are:

$$\exists\, \tilde{n} \neq 0 \;\Leftrightarrow\; \boldsymbol{A}^T\,(q \otimes E)\,n = n \;\vee\; \boldsymbol{A}^T\,(E \otimes q)\,n = n \tag{4.2}$$

If the rectangular matrix $\boldsymbol{A}$, which represents the K-algebra, has full rank then these systems of equations are only solvable for certain q ('local' zero-states).

It is obvious to link local zero-states with black holes, due to similar behaviour. The singularity of a black hole "eats up" every other state, just like the zero-state. The difference is in the mightiness. But if a black hole cannot collapse to a point and only to a minimal mass shell, then the forces in the centre of the mass shell would cancel each other out exactly and become zero - a local zero-state in the quantum view?

ALGEBRAIC VACUUM-STATE

The constraints of existence for a right-handed 'one' for any K-algebra are as follows:

$$\forall \tilde{q} \neq 0 \;\wedge\; \tilde{o}_{\mathbb{A}} \neq 0 \;\Rightarrow\; \tilde{q} \otimes_{\mathbb{A}} \tilde{o}_{\mathbb{A}} = \tilde{q} \qquad \tilde{o}_{\mathbb{A}} = \beta^T o_{\mathbb{A}} \tag{4.3}$$

This problem can be reformatted to an overdetermined matrix vector standard problem in the RKS notation using the Neudecker function and a Kronecker/shift operator identity (A.3/F.5). The advantage of the RKS notation over an index notation emerges here.

The following RKS formula now outputs any 'one' for all K-algebras, if the algebra follows certain constraints (third line, right side):

$$\begin{aligned}
&\boldsymbol{A}^T\,(E \otimes o)q = q \;\Leftrightarrow\; \boldsymbol{A}^T\,(E \otimes o) = E \;\Rightarrow\\
&vec\big(\boldsymbol{A}^T\,(E \otimes o)\big) = \boldsymbol{A}^{BR} o = vec(E) \;\Rightarrow\\
&o_{\mathbb{A}} = \boldsymbol{A}^{BRI} vec(E) \;\Leftrightarrow\; \boldsymbol{A}^{BR}\boldsymbol{A}^{BRI} vec(E) = vec(E)
\end{aligned} \tag{4.4}$$

The right-handed one is unique and minimal, if there is no kernel solution for $\boldsymbol{A}^{BR}$ i.e. $\boldsymbol{A}^{BR}$ has full rank. The solution for the left-handed one is structurally the same, here the algebra must be replaced by the bilaterally shifted one ($\boldsymbol{A}^{BRI} \rightarrow \boldsymbol{A}^{BBRI} = \boldsymbol{A}^{RI}$).

If right- and left-handed 'one' both exist together, then they are necessarily identical. The quantum mechanical vacuum state with the usual symbol $|0\rangle$ is here identified with a domain dependent algebraic field $o_{\mathbb{A}}(q)$, since the 'one' is neutral to any other state. Three different handed vacuum states are possible locally (right-, left- or both-handed) but also local regions in the algebraic field that do not have a vacuum state are allowed.

This opens up a new understanding of particles and fields. Fields must exist in a vacuum and therefore have a local vacuum state. Particles are then enclosed regions without a vacuum state. The changing boundaries between regions without and with vacuum state can then represent higher particle states in this new algebraic image. The following section shows that the existence of a conjugation (dual space) depends on the existence of the vacuum state.

5. ALGEBRAIC DUAL SPACE

From the basis invariant definition (2.13) following properties are valid for the operator $H_{\mathbb{A}}$:

$$H_{\mathbb{A}}{}^{T} = H_{\mathbb{A}} \quad \text{and} \quad H_{\mathbb{A}}^{2} = E \tag{5.1}$$

To determine the minimal Hermitian operator for any K-algebra, the scalar product of the dual base is formed with the original base. This results in the algebraic one, if the K-algebra has a one (4.4) otherwise no conjugation is possible.

$$\beta^{I}\,\beta = \tilde{o}_{\mathbb{A}} \tag{5.2}$$

This has its physical equivalent, because particles and anti-particles can only be generated from a vacuum (=algebraic one) if a vacuum state exists in the algebraic primeval field.

$$\mathrm{m}_{\mathbb{A}}^{2}(\beta)\,\tilde{o}_{\mathbb{A}} = \beta^{T} H_{\mathbb{A}}\,\beta = vec(H_{\mathbb{A}})^{T}(\beta \otimes_{\mathbb{A}} \beta) = vec(H_{\mathbb{A}})^{T}\boldsymbol{A}\,\beta \tag{5.3}$$

Using Kronecker/shift operator Symmetries (A.2/A.3), the following results are obtained:

$$H_{\mathbb{A}} = \mathrm{m}_{\mathbb{A}}^{2}\,\boldsymbol{A}^{IRR}(o_{\mathbb{A}} \otimes E) \iff \boldsymbol{A}^{IT}\boldsymbol{A}^{T} vec(H_{\mathbb{A}}) = vec(H_{\mathbb{A}}) \tag{5.4}$$

The missing metric coefficient can be determined by the involution (5.1).

This requires following relationship for the K-algebra for a real domain:

$$\boldsymbol{A}^{I} = \boldsymbol{A}^{T}\mathrm{m}_{\mathbb{A}}^{-2} \tag{5.5}$$

Thus, the metric coefficient can be determined with the help of the trace of a matrix:

$$\mathrm{m}_{\mathbb{A}}^{2} = \frac{trace(\boldsymbol{A}^{T}\boldsymbol{A})}{trace(E)} \tag{5.6}$$

The searched for operator is then given under the conditions (4.4) of the existence of a one and under the following additional constraints:

$$H_{\mathbb{A}} = \boldsymbol{A}^{RRT}(o_{\mathbb{A}} \otimes E) \Leftrightarrow \boldsymbol{A}\,\boldsymbol{A}^{T} vec(H_{\mathbb{A}}) = \mathrm{m}_{\mathbb{A}}^{2}\, vec(H_{\mathbb{A}}) \tag{5.7}$$

It is worth to note, that the here defined operator is unique and minimal, because of the underdetermined equation system (5.3) and will differ in some cases to given conjugation operators in literature which are defined by mimicking the complex numbers. This minimal principle is used as a guide line for uniqueness of all operators in Algebraic Field Theory.

6. ALGEBRAIC DECAY OF THE VACUUM

For a two-handed algebraic decay of the vacuum state shall apply:

$$\tilde{q}^{I} \otimes_{\mathbb{A}} \tilde{q} = \tilde{o}_{\mathbb{A}} = \tilde{q} \otimes_{\mathbb{A}} \tilde{q}^{I} \tag{6.1}$$

Writing this out requires the following relationship to be valid to fulfil it:

$$\forall q \neq 0: \quad \boldsymbol{A}^{T}(H_{\mathbb{A}} \otimes E - E \otimes H_{\mathbb{A}})(q \otimes q) = 0 \tag{6.2}$$

A solution is given if the K-algebra is subject to the following constraint:

$$\forall q, \boldsymbol{X} \neq 0: \ \boldsymbol{X}^{T}(q \otimes q) = 0 \Leftrightarrow \boldsymbol{X} = \boldsymbol{Y} - \boldsymbol{Y}^{B} \tag{6.3}$$

It follows with that:

$$(H_{\mathbb{A}} \otimes E)\,\boldsymbol{A} = ((E \otimes H_{\mathbb{A}})\boldsymbol{A})^{B} \tag{6.4}$$

Off (6.1) then applies to the searched operator *G*:

$$\boldsymbol{A}^T(H_{\mathbb{A}} \otimes E)(q \otimes q) = \ (q^T \ G \ \ q) \ o_{\mathbb{A}} = vec(G)^T(q \otimes q) \ o_{\mathbb{A}} \tag{6.5}$$

A structural comparison between the right and the left side then shows using Kronecker/shift operator symmetries (A.2/A.3) that the following identities must be valid:

$$(H_{\mathbb{A}} \otimes E) \ \boldsymbol{A} \ {o_{\mathbb{A}}}^{IT} = vec(G) = vec(\boldsymbol{A}^{RRT}({o_{\mathbb{A}}}^{IT} \otimes H_{\mathbb{A}})) \tag{6.6}$$

Thus, the K-algebra also determines the minimal metric for the vacuum state, if a 'one' and a minimal conjugation are present locally and under the following additional conditions:

$$G_{Vacuum} = \ \boldsymbol{A}^{RRT}({o_{\mathbb{A}}}^{IT} \otimes H_{\mathbb{A}}) \ \Leftrightarrow \ (H_{\mathbb{A}} \otimes E) \ \boldsymbol{A} \ = ((E \otimes H_{\mathbb{A}})\boldsymbol{A})^B \tag{6.7}$$

Theoretically, one of the two sides can be dropped from the conditions in (6.1). This would then generally lead to two different chiral metric operators. As for the minimal conjugation, the minimal metric operator can differ from given regular metrics.

7. ALGEBRAIC ANALYSIS

In the introduction it was already mentioned, that due to common properties the mathematical structures of quantum theory and general relativity can be derived from algebraic constraints of the binary operation. The complex analysis with the Cauchy-Riemannian constraints shall serve as a model here.

Analogous to the complex procedure, the following pseudo-linear functional shall apply:

$$\tilde{l}_{\mathbb{A}}(q,p)[f] \ = \tilde{q} \ \ \otimes_{\mathbb{A}(q,p)} \ \tilde{p}[f] \tag{7.1}$$

For the complex numbers, the complex derivation $\tilde{p}$ results as a special case from the infinitesimal difference of this functional in the representation by $\tilde{q}$.

$$d \ \tilde{l}_{\mathbb{C}}(q)[f] = d\tilde{q} \ \ \otimes_{\mathbb{C}} \ \tilde{p}[f] \ = d\tilde{f}(\tilde{q}) \tag{7.2}$$

This relation is only valid for constant K-algebras and if the derivative itself is extreme ($d\tilde{p}[f] = 0$). For an algebraic field, the complete total differential must apply over the entire product including the algebra:

$$d \ \tilde{l}_{\mathbb{A}}(q,p)[f] = d \ (\ \tilde{q} \ \ \otimes_{\mathbb{A}(q,p)} \ \tilde{p}[f] \ ) \tag{7.3}$$

Furthermore, for the searched, pseudo-linear operators it is required that the total infinitesimal difference is independent of the order of the operators in the functional:

$$d \ \tilde{l}_{\mathbb{A}}(q,p)[f] = d \ \tilde{l}_{\mathbb{A}}(p,q)[f] \tag{7.4}$$

This is justified by the linearity requirements, because for comparison this property also applies to the commutator of the standard derivation (and later also to the quantum mechanical commutator).

$$\left[\partial_j, q^k\right] = \ \delta_j^k \ \ \Rightarrow \ \ d\left[\partial_j, q^k\right] = \ 0 \tag{7.5}$$

A quantum mechanical matrix view according to Werner Heisenberg now helps to find a solution in closed form for this algebraic problem. Two operators *P*, *Q* are interpreted as a quadratic matrix with the following properties:

$$\beta^T P \ f \ \coloneqq \ \tilde{p}[f] \ \ \text{and} \ \ \ Q \coloneqq \ \boldsymbol{A}^T(q \otimes E) \tag{7.6}$$

P plays the role of the energy impulse tensor (here the to be determined algebraic differential operator) and *Q* the role of the algebraic local operator.

With the help of the operator (2.12), the problem (7.3) can be reformulated into a form, free of the generating base as follows:

$$\boldsymbol{\nabla}(QPf) = \boldsymbol{\nabla} f \text{ by } \boldsymbol{\nabla}\beta = 0 \tag{7.7}$$

The left side then results in the following relation, where the index c at an operator means that this is to be treated as a constant, with respect to the action of the first differential operator to the left of it.

$$\boldsymbol{\nabla}(QPf) = \boldsymbol{\nabla}(Q)Pf + \boldsymbol{\nabla}Q_c(Pf) \tag{7.8}$$

From the validity of (7.4) follows simultaneously:

$$\boldsymbol{\nabla} f = \boldsymbol{\nabla}(P(Q_c f)) \tag{7.9}$$

If equation (7.9) is subtracted from (7.8), the commutator brackets are used:

$$\boldsymbol{\nabla}([Q_c, P]f) + \boldsymbol{\nabla}(Q)Pf = 0 \tag{7.10}$$

If you add both equations (7.9) and (7.8), using the anti-commutator brackets, the following is shown:

$$\boldsymbol{\nabla}(\{Q_c, P\}f) + \boldsymbol{\nabla}(Q)Pf = 2\,\boldsymbol{\nabla} f \tag{7.11}$$

Since the operator P must be pseudo-linear as a differential operator, this only works if the following additional condition applies, with a rectangular matrix operator $\boldsymbol{J}$, which has yet to be determined:

$$\boldsymbol{\nabla}(\{Q_c, P\}f) = (2\,\boldsymbol{J} - \boldsymbol{\nabla}(Q))Pf \tag{7.12}$$

This reduces the problem to a known, overdetermined and pseudo-linear system:

$$\boldsymbol{\nabla} f = \boldsymbol{J}\, P\, f \tag{7.13}$$

The solution is then subject to generalized Cauchy-Riemann constraints:

$$P = \boldsymbol{J}^I\, \boldsymbol{\nabla} \;\Leftrightarrow\; \boldsymbol{J}\,\boldsymbol{J}^I \boldsymbol{\nabla} f = \boldsymbol{\nabla} f \tag{7.14}$$

To determine the operator $\boldsymbol{J}$, first consider the following derivation of the general commutator between Q and P:

$$\boldsymbol{\nabla}([Q, P]f) = \boldsymbol{\nabla}(Q)Pf - \boldsymbol{\nabla}(P(Q)f) + \boldsymbol{\nabla}([Q_c, P]f) \tag{7.15}$$

This equation is reduced because of (7.10) to the following relation:

$$\boldsymbol{\nabla}([Q, P]f) = -\boldsymbol{\nabla}(P(Q)f) \tag{7.16}$$

For this to be fulfilled, the following must apply to the commutator:

$$[P, Q] = P(Q) \tag{7.17}$$

By (7.5):

$$P(Q) = E \tag{7.18}$$

At the same time, the searched for operator $\boldsymbol{J}$ is already fixed, provided that it has full rank:

$$\boldsymbol{J} = \boldsymbol{\nabla} Q \;\Leftrightarrow (\boldsymbol{\nabla} Q)^I\, \boldsymbol{\nabla} Q = E \tag{7.19}$$

The minimal, algebraic-affine derivative operator can thus be specified completely. It is subject to three algebraic constraints for existence:

$$P = (\boldsymbol{\nabla} Q)^I \, \boldsymbol{\nabla} \qquad \Leftrightarrow \left((\boldsymbol{\nabla} Q)(\boldsymbol{\nabla} Q)^I - E \otimes E\right) \boldsymbol{\nabla} f = 0 \tag{7.20}$$

$$\wedge \quad \boldsymbol{\nabla} \left(Q_c (\boldsymbol{\nabla} Q)^I \, \boldsymbol{\nabla} f\right) = 0 \quad \wedge \quad (\boldsymbol{\nabla} Q)^I \, \boldsymbol{\nabla} Q - E = 0$$

Due to the symmetries of the shift operators, a total of six different variants are possible for the P operator. These can be grouped into three related families $\{\boldsymbol{A},\ \boldsymbol{A}^B\}, \{\boldsymbol{A}^R,\ \boldsymbol{A}^{RB}\}, \{\boldsymbol{A}^{RR}, \boldsymbol{A}^{BR}\}$ according to chirality.

The first overdetermined constraints in the system (7.20) are the generalization of the Cauchy-Riemann equations, because they form a Dirac-Maxwell system in the limit value of vanishing gravity ($\boldsymbol{\Gamma} = 0$ and thus $\partial \boldsymbol{A} = 0$). Therefore, Algebraic Field Theory is structurally transferred into quantum theory when gravity disappears. This means that Algebraic Field Theory introduces gravity into quantum theory at the expense of an additional local field of algebraic nature which change can be interpreted as curvature ($\partial \boldsymbol{A} \sim \boldsymbol{\Gamma}$). However, in Algebraic Field Theory there are 3x2 different versions due to the symmetry of the shift operators (7.21, the left version as an example), i.e. the Dirac equation has five more siblings with respective additional constraints. Their physical significance must be left open at this early stage of theory. Also, the change of the local algebraic field will introduce negative and positive modes to curvature. Negative modes or negative gravity had never been observed beside the effects of dark energy which the experimental data are still in discussion.

$$P_0 = \boldsymbol{A}^{RI} (\partial \otimes E) \ = \ \boldsymbol{A}^{RI} \left(e_{\mathrm{j}} \otimes E\right) \partial_{\mathrm{j}} \tag{7.21}$$

$$\Leftrightarrow \ \boldsymbol{A}^R \boldsymbol{A}^{RI} (\partial \otimes E) f = (\partial \otimes E) f$$

If the rectangular matrix which represents the complex numbers (see section introduction 1.2, see also A.5) is taken into the constraint (7.21), the classical Cauchy-Riemann differential equations are reproduced. Equation (7.21) can now also be applied to all other quantum algebras, e.g. the K-algebra of Gell-Mann's λ matrices. These are now also subject to additional conditions (A.8). Constraint (7.21, line 2) is also applicable for K-algebras without algebraic 'one', e.g. the standard three-dimensional cross product between vectors (see A.6, generalized Cauchy-Riemann equations for the cross product).

The second additional constraint in (7.20) is a mixed differential equation system of the second order, sharing the order of the classical vacuum equation of general relativity but not identical. This additionally restricts the solution space of the first constraint and is an outgoing point to falsify either classic general relativity in the quantum realm or this theory.

So far, the operator P is not yet a physical energy impulse operator, but only a purely generic algebraic operator. Equation (7.18) has the degree of freedom of free choice of a constant for ensuring pseudo linearity. In order to move now to a specific Algebraic Field Theory as a variant, the constant can be replaced by the usual quantum mechanical counterpart using here the Minkowski metric G_0.

$$\left[\partial_{\mathrm{j}}, q^k\right] = \ \mathrm{i}\,\hbar\, \mathrm{G}_0{}_j^k \quad \text{with} \quad G_0 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \tag{7.22}$$

With (7.22) the defining equation (7.18) becomes $P_{Physics}$ for the "physical" operator:

$$P_{Physics}(Q) = \mathrm{i}\,\hbar G_0 \;\Rightarrow\; P_{Physics} = \mathrm{i}\,\hbar\, G_0\,(\boldsymbol{\nabla} Q)^I\,\boldsymbol{\nabla} \tag{7.23}$$

This describes an "algebraic quantum computer" as a specific variant.

So where does the fundamental equation (7.22) for quantum theory originates from?

This may be related to the type of expansion of the vacuum and the associated position of the observer. Suppose the universe is closed in itself in the form of a corresponding S^3-sphere, which itself is located in a four-dimensional Euclidean space plus time. The 'surface' (hyperface) of this S^3-sphere is the metric, three-dimensional space and time forms a common vector from the origin to the surface of the S^3-sphere. If now this surface expands outward with the limit speed (at the speed of light), all observers within this expanding surface would not notice the additional dimension at all on a large scale, since interactions can only work perpendicular to the direction of expansion, i.e. only within the S^3 surface. The observers on the surface then do not discover the Euclidean metric as a conservation quantity, but the Minkowski metric and from its divisional algebra the `i` of the complex numbers is also needed. From the point of view of the origin, the universe would satisfy the following Euclidean relationship with an additional fourth spatial dimension q^4.

$$(\mathrm{ct})^2 = (\mathrm{q}^1)^2 + (\mathrm{q}^2)^2 + (\mathrm{q}^3)^2 + (\mathrm{q}^4)^2 \tag{7.24}$$

From the view of the observers, the fourth dimension of space will be noticeable as "dents" $(\delta \mathrm{q}^4)^2$ in their space-time, caused by local material distortions into the fourth dimension of space and thus for all observers in the spherical surface always a conservation quantity. From that follows the Minkowski metric, if one now includes time in the metric:

$$(\delta \mathrm{ct})^2 - (\delta \mathrm{q}^1)^2 - (\delta \mathrm{q}^2)^2 - (\delta \mathrm{q}^3)^2 = (\delta \mathrm{q}^4)^2 = \mathrm{G}_{0\mathrm{jk}} \mathrm{x}^{\mathrm{j}} \mathrm{x}^{\mathrm{k}} \text{ by } x = (\delta \mathrm{ct}, \delta q) \tag{7.25}$$

To sum up, the Minkowski metric and the quantum relation (7.22) might originate from the fact that we ride like surfers on a wave which propagates at the speed of light in a higher dimensional Euclidean space (in the IT picture an algebraic wave at maximum processor speed in the memory). The physical *P*-operator (7.23) then applies only specifically to the wave but no longer in general!

Dirac-Operator in comparison to the physical *P*-Operator at vanishing gravity (7.21):

$$P_{Dirac} = \mathrm{i}\,\hbar\;\gamma^{\mathrm{j}}\,\partial_{\mathrm{j}} \quad \leftrightarrow \quad P_{Physics}\,(\boldsymbol{\Gamma} = 0) = \mathrm{i}\,\hbar\, G_0\, \boldsymbol{A}^{RI}\left(e_{\mathrm{j}} \otimes E\right)\partial_{\mathrm{j}} \tag{7.26}$$

This requires the validity of the following relationship:

$$\gamma_{\mathrm{j}} = \boldsymbol{A}^{RI}\left(e_{\mathrm{j}} \otimes E\right) \tag{7.27}$$

By means of reformatting and application of Kronecker/shift operator symmetries (A.2/A.3), the related K-algebra for the Dirac's gamma matrices (A.7) follow.

$$\boldsymbol{A} = \left(\left(e_{\mathrm{j}} \otimes E\right)\gamma_{\mathrm{j}}\right)^{BRITRR} \tag{7.28}$$

Now, one faces a mathematical issue, because the gamma matrices only form a complete generating system with 16 independent 4x4 matrices. The Dirac equation itself is only developed over a four-dimensional subspace of the first four gamma matrices. In four space-time dimensions the K-algebra (A.7), recalculated by means of (7.28), has no algebraic unity and thus no vacuum state.

This corresponds to the algebraic picture, because the Dirac equation describes an elementary particle without external field, which now in the algebraic view can no longer decay, because a vacuum state would be necessary for this. The application of the relation (6.7) for the metric is therefore no longer permitted, since it is only valid for an existing vacuum state. For the metric of a particle induced by the algebra, the Dirac condition (1.5) is still available, because this does not require a vacuum state. With (7.27) the following relation can be established for the 'inner' metric of a particle without a vacuum state:

$$G_{Particle} = \tfrac{1}{2}\ Trace(\{G_0\boldsymbol{A}^{RI}(e_j \otimes E), G_0\boldsymbol{A}^{RI}(e_{\mathrm{k}} \otimes E)\})\ (e_{\mathrm{j}} \otimes e_k^T) \tag{7.29}$$

Inside an elementary particle which can't decay, there is an algebraic as well as a metric field in the same way as outside. Unlike on the outside there is no vacuum state inside an elementary particle. Only the boundary of the elementary particle is measurable, because without a vacuum state there is no zero-point reference as a prerequisite for a comparable measurement.

8. VACUUM EQUATION

All means are now available for calculating the energy/mass k of a physical vacuum solution. The following eigenvalue vacuum equation applies to $P_{Physics}$:

$$\mathrm{i}\ G_0(\boldsymbol{\nabla}Q_{\mathbb{A}})^I\ \boldsymbol{\nabla}o_{\mathbb{A}} = \mathrm{k}\ o_{\mathbb{A}} \tag{8.1}$$

In order to solve this equation, all permitted algebraic fields must be determined. First, all previously determined constraints for the algebraic field must be collected. A two-handed vacuum must therefore fulfil all following eight systems (3.16, 4.4[2], 5.7, 6.7, 7.20[3]). All occurring factors can be expressed by the K-algebra. Topological reasons[16] do limit non-trivial algebraic solutions to the finite degrees of freedom N=2,4,8 in general for these eight systems, since the K-algebras must be composite, real divisional algebras according to the conditions for the vacuum. Since only a physical solution is sought, only N=4 applies but also N=2 or N=8 and combinations out of them are possible.

Of the eight systems, four are differential equations:

$$\left(\frac{\partial \boldsymbol{A}}{\partial q}\right) = -\boldsymbol{A}\ ((\partial\ \otimes G)^T + (\partial\ \otimes G)^{RT} - (\partial\ \otimes G)^{RRT})(E \otimes G^I) \tag{8.2}$$

$$\left((\boldsymbol{\nabla}Q)(\boldsymbol{\nabla}Q)^I - E \otimes E\right)\boldsymbol{\nabla}o_{\mathbb{A}} = 0$$

$$\boldsymbol{\nabla}(\{Q_c, \mathrm{i}G_0(\boldsymbol{\nabla}Q)^I\boldsymbol{\nabla}\}o_{\mathbb{A}}) = (\boldsymbol{\nabla}(Q)(\mathrm{i}\ G_0 - 2E)(\boldsymbol{\nabla}Q)^I\boldsymbol{\nabla}o_{\mathbb{A}}$$

$$(\boldsymbol{\nabla}Q)^I\ \boldsymbol{\nabla}Q - E\ = 0$$

The remaining four systems further restrict the initial conditions and solution space:

$$(H_{\mathbb{A}} \otimes E)\ \boldsymbol{A}\ = \left((E \otimes H_{\mathbb{A}})\boldsymbol{A}\right)^B \tag{8.3}$$

$$\boldsymbol{A}\ \boldsymbol{A}^T vec(H_{\mathbb{A}}) = \mathrm{m}_{\mathbb{A}}^2\ vec(H_{\mathbb{A}})$$

$$\boldsymbol{A}^{BR}\boldsymbol{A}^{BRT} vec(E) = \mathrm{m}_{\mathbb{A}}^2\ vec(E)$$

$$\boldsymbol{A}^{R}\boldsymbol{A}^{RT} vec(E) = \mathrm{m}_{\mathbb{A}}^2\ vec(E)$$

A solution of the intricate systems (8.1+8.2 & 8.3) is a challenge and are requiring a significant amount of computer power. An open source implementation of the Algebraic Field Theory is attached to this script to demonstrate the feasibility and practical application, including a solution at least for the (8.3) system in two dimensions which took already several hours for the calculation on a PC.

A discussion of various solution strategies would clearly exceed the scope of this work, intended as an introduction. For finding solutions the generalizations of the Cauchy integral formulas to arbitrary K-algebras are helpful. With the new operators presented here, this is possible by starting from the definition of an integral via the complex numbers, and applies analogously to Algebraic Field Theory:

$$\int d\,(\tilde{q} \otimes_{\mathbb{A}} \tilde{p}[f]) = \tilde{f}(\tilde{q}) \tag{8.4}$$

Summarizing, the primary goals of this script are the demonstration of the potential of an Algebraic Field Theory and that with the few required axioms, the structures of quantum theory and at least similar structures to general relativity can be derived from the principle of a local changing binary operation. A quantum gravity theory of the vacuum itself is established and at the same time the experimental possibility of falsifying algebraic fields is demonstrated by measuring the spin of highly accelerated electrons in orbits. Vice versa, if any experiment detects a spin dependency by acceleration then standard quantum theory is falsified in the sense of Popper and would require an algebraic extension to include gravity.

The author assumes from first calculations, that the operator $P_{Physics}$ is bounded downward and upward. That would mean for the vacuum a minimum mass or energy in a maximum possible, finite expansion or vice versa only a maximum mass or energy is possible in the minimum expansion. This can be taken rudimentarily from the structures. Because of the fourth line in (8.2) $\boldsymbol{\nabla} Q$ can never disappear. Then the term in (8.1) must be $\boldsymbol{\nabla} o_{\mathbb{A}} = 0$. This relationship follows the definition of the local frame (2.12) and leads to contradictions in (8.2). The first and third line seem to require a finite $\boldsymbol{\nabla} o_{\mathbb{A}} \neq 0$.

"The quantum phenomenon seems to indicate with certainty that a finite system of finite energy can be completely described by a finite number of numbers (quantum numbers).

This must lead to an attempt to describe reality by a purely algebraic theory.

But no one sees how the basis of such a theory could be obtained!"

Albert Einstein

Ω

APPENDIX

A.1) Cartesian product of K-algebras

The numbers for the domain $\mathbb{K}$ should now also follow the structure of (2.3) with their own generators. Thus, they are no longer necessarily a field, but at least they should have the real numbers themselves as inner subspace. Be now $\beta_{\mathbb{A}}$, $\beta_{\mathbb{K}}$ the respective bases of the generators for the K-algebras $\mathbb{A}$ and $\mathbb{K}$.

The cartesian product K-Algebra $\mathbb{A}'$, in respect to the base $\beta_{\mathbb{K}} \otimes \beta_{\mathbb{A}}$ is:

$$(\beta_{\mathbb{K}} \otimes \beta_{\mathbb{A}}) \otimes_{\mathbb{A}'} (\beta_{\mathbb{K}} \otimes \beta_{\mathbb{A}}) = \boldsymbol{A}' (\beta_{\mathbb{K}} \otimes \beta_{\mathbb{A}})$$

By (2.3) the two bases must commutate by definition. In addition, an operator 'B' is required, to swap the factors in the Kronecker product. 'B' is a generalization of the definition (2.6) for two different dimensional bases.

$$\beta_{\mathbb{K}} \otimes (\beta_{\mathbb{A}} \otimes \beta_{\mathbb{K}}) \otimes \beta_{\mathbb{A}} = \beta_{\mathbb{K}} \otimes \big(B\,(\beta_{\mathbb{K}} \otimes \beta_{\mathbb{A}})\big) \otimes \beta_{\mathbb{A}}$$

$$B \coloneqq \Sigma_j \left(e_j(Dim\ \beta_{\mathbb{A}}) \otimes E(Dim\ \beta_{\mathbb{K}}) \otimes e_j^T(Dim\ \beta_{\mathbb{A}})\right)$$

With the help of standard Kronecker identities this results:

$$(E \otimes B)\,(\beta_{\mathbb{K}} \otimes \beta_{\mathbb{K}} \otimes \beta_{\mathbb{A}}) \otimes \beta_{\mathbb{A}} = (E \otimes B \otimes E)\,(\boldsymbol{K}\beta_{\mathbb{K}} \otimes \boldsymbol{A}\beta_{\mathbb{A}}) = (E \otimes B \otimes E)(\boldsymbol{K} \otimes \boldsymbol{A})\,(\beta_{\mathbb{K}} \otimes \beta_{\mathbb{A}})$$

In summary, any combined complex structure (e.g. bi-quaternions) can be mapped to a higher dimensional K-algebra with the real numbers as field. The corresponding K-algebra is then given:

$$\boldsymbol{A}' = \left(E_{Dim(\beta_{\mathbb{K}})} \otimes B_{Dim(\beta_{\mathbb{K}} \otimes \beta_{\mathbb{A}})} \otimes E_{Dim(\beta_{\mathbb{A}})}\right)(\boldsymbol{K} \otimes \boldsymbol{A})$$

A.2) Shift Operator relations für B, I, R, T

$$BRB = RR \qquad BRR = RB \qquad RRB = BR \qquad RBR = B$$

$$BB = II = RRR = TT = E$$

$$[T, B] = [T, I] = [T, R] = [B, I] = 0$$

$[B, R]$ und $[I, R]$ do not commute generally!

A.3) Kronecker Shift Operator Identities

The listed identities can be derived from the shift operator definitions in combination with Kronecker product identities.

F.1 $(\partial \otimes E)(\boldsymbol{A}^T(q \otimes E)) = \boldsymbol{A}^{BR} \Leftrightarrow (\partial \otimes E)\boldsymbol{A}^T = 0$

F.2 $(q^T \otimes E)\boldsymbol{A}^{BR} = \boldsymbol{A}^T(q \otimes E)$

F.3 $\beta^T\boldsymbol{A}^T(p \otimes q) = p^T\boldsymbol{A}^{RT}(q \otimes \beta) = q^T\boldsymbol{A}^{RRT}(\beta \otimes p) = \beta^T\boldsymbol{A}^{BT}(q \otimes p) = q^T\boldsymbol{A}^{BRT}(p \otimes \beta) = p^T\boldsymbol{A}^{RBT}(\beta \otimes q)$

F.4 $p^T\boldsymbol{A}^{BRT}(Q \otimes q)q = q^T\boldsymbol{A}^T(Q \otimes p)\,q$

F.5 $vec(\boldsymbol{A}^T\,(q \otimes Q)) = (Q^T \otimes E)\boldsymbol{A}^R q$

F.6 $\boldsymbol{A}^{RRT}(E \otimes q) = \Sigma_{\mathrm{j}}[\boldsymbol{A}]_{\mathrm{j}}\,q_{\mathrm{j}}$ with $[\boldsymbol{A}]_{\mathrm{j}} \coloneqq \boldsymbol{A}^T\left(e_{\mathrm{j}} \otimes E\right)$ (the j-th sub matrix from top)

F.7 $(\boldsymbol{A}^T(p \otimes q))_{\mathrm{j}} = p^T[\boldsymbol{A}^{RR}]_{\mathrm{j}}\,q$

F.8 $(E \otimes \boldsymbol{A})\boldsymbol{A} = (\boldsymbol{A} \otimes E)\boldsymbol{A} \quad \Leftrightarrow$ $\boldsymbol{A}$ is associative

F.9 $(\boldsymbol{A} - \boldsymbol{A}^B + \boldsymbol{A}^R - \boldsymbol{A}^{BR} + \boldsymbol{A}^{RR} - \boldsymbol{A}^{RB}) = \Sigma_{\mathrm{jkl}}\left(\mathrm{A}_{\mathrm{jkl}}\varepsilon_{\mathrm{jkl}}\right)\boldsymbol{\mathcal{E}}$ (Levi-Civita Tensor)

A.4) Local Frame

$$(G\nabla) \otimes \beta = e_{\mathrm{j}} \otimes (\mathrm{G}^{\mathrm{jl}}\nabla_{\mathrm{l}}\, e_{\mathrm{k}})\beta^{\mathrm{k}} = \Gamma_{\mathrm{l}}^{\mathrm{jk}}\beta^{\mathrm{l}}(e_{\mathrm{j}} \otimes e_{\mathrm{k}}) = \Gamma_{\mathrm{l}}^{\mathrm{jk}}(e_{\mathrm{j}} \otimes e_{\mathrm{k}} \otimes e^{\iota})e_{\mathrm{l}}\beta^{\mathrm{l}}) = \boldsymbol{\Gamma}\,\beta$$

A.5) *Standard complex numbers* $\mathbb{C}$

$$\beta = \begin{pmatrix}1\\ \mathrm{i}\end{pmatrix} \qquad \boldsymbol{A} = \begin{pmatrix} a_1^{11} & a_2^{11} \\ a_1^{12} & a_2^{12} \\ a_1^{21} & a_2^{21} \\ a_1^{22} & a_2^{22} \end{pmatrix} := \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ -1 & 0 \end{pmatrix}$$

$$\boldsymbol{A}^R = \begin{pmatrix} a_1^{11} & a_1^{21} \\ a_2^{11} & a_2^{21} \\ a_1^{12} & a_1^{22} \\ a_2^{12} & a_2^{22} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & -1 \\ 1 & 0 \end{pmatrix} \Rightarrow \quad \boldsymbol{A}^{RI} = \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 \end{pmatrix}$$

$$\Rightarrow \; P_{\mathbb{C}} := \boldsymbol{A}^{RI}(\partial \otimes E) = \frac{1}{2}\begin{pmatrix} \partial_1 & \partial_2 \\ -\partial_2 & \partial_1 \end{pmatrix}$$

$$\Leftrightarrow (E \otimes E - \boldsymbol{A}^R\boldsymbol{A}^{RI})(\partial \otimes E)f = 0$$

$$= \begin{pmatrix} \partial_1 f_1 - \partial_2 f_2 \\ \partial_1 f_2 + \partial_2 f_1 \\ \partial_2 f_1 + \partial_1 f_2 \\ \partial_2 f_2 - \partial_1 f_1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \text{ (Cauchy-Riemann)}$$

$$H_{\mathbb{C}} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad G_{\mathbb{C}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad o_{\mathbb{C}} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

A.6) *Vector cross product*

K-Algebra $\boldsymbol{\mathcal{E}}$ (Levi-Civita Tensor) (Symbol $\times$)

$$\boldsymbol{\mathcal{E}}^T{:}\times = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\Rightarrow \frac{1}{2}\boldsymbol{\mathcal{E}}^{RI} = -\frac{1}{2}\boldsymbol{\mathcal{E}}^{BRI}$$

Here two-handed *P* operators exist:

$$\Rightarrow \; P_r{:}\times = \frac{1}{2}\begin{pmatrix} 0 & \partial_3 & -\partial_2 \\ -\partial_3 & 0 & \partial_1 \\ \partial_2 & -\partial_1 & 0 \end{pmatrix} \qquad P_l{:}\times = -P_r{:}\times$$

The generalized Cauchy-Riemann constraints are identical for both operators:

$$\Leftrightarrow CR_r = CR_l = \begin{pmatrix} \partial_1 f_1 = \partial_2 f_2 = \partial_3 f_3 = 0 \\ \partial_1 f_2 + \partial_2 f_1 = \partial_1 f_3 + \partial_3 f_1 = \partial_2 f_3 + \partial_3 f_2 = 0 \end{pmatrix} = (\partial \otimes E + E \otimes \partial)f = 0$$

The algebraic one does not exist, so conjugation is not possible.

A.7) *Dirac K-Algebra*

The following representation of Dirac's gamma matrices is used:

$$\gamma^1 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \quad \gamma^2 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix} \quad \gamma^3 = \begin{pmatrix} 0 & 0 & 0 & -\mathrm{i} \\ 0 & 0 & \mathrm{i} & 0 \\ 0 & \mathrm{i} & 0 & 0 \\ -\mathrm{i} & 0 & 0 & 0 \end{pmatrix} \quad \gamma^4 = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

The K-Algebra $\boldsymbol{D}_4$ results then by (7.24):

$$\boldsymbol{D}_4^T = \frac{1}{2i}\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & \mathrm{i} & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 1 & -\mathrm{i} & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & -1 & -\mathrm{i} & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -\mathrm{i} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

For the associated Dirac Operator,

$$P_0 = \mathrm{i}\,\gamma^{\mathrm{k}}\partial_{\mathrm{k}}$$

following Cauchy Riemann constraints result, where the listed vector is to be set equal to the corresponding zero vector:

$$\begin{pmatrix} \partial_1\varphi_1 - \partial_4\varphi_3 - (\partial_2 - \mathrm{i}\,\partial_3)\varphi_4 \\ \partial_1\varphi_2 - \partial_2\varphi_3 - \mathrm{i}\,\partial_3\varphi_3 + \partial_4\varphi_4 \\ -\partial_4\varphi_1 - \partial_2\varphi_2 + \mathrm{i}\,\partial_3\varphi_2 + \partial_1\varphi_3 \\ -\partial_2\varphi_1 - \mathrm{i}\,\partial_3\varphi_1 + \partial_4\varphi_2 + \partial_1\varphi_4 \\ \partial_2\varphi_1 - \mathrm{i}\,\partial_3\varphi_1 + \partial_4\varphi_2 - \partial_1\varphi_4 \\ -\partial_4\varphi_1 + \partial_2\varphi_2 + \mathrm{i}\,\partial_3\varphi_2 - \partial_1\varphi_3 \\ -\partial_1\varphi_2 + \partial_2\varphi_3 - \mathrm{i}\,\partial_3\varphi_3 + \partial_4\varphi_4 \\ -\partial_1\varphi_1 - \partial_4\varphi_3 + (\partial_2 + \mathrm{i}\,\partial_3)\varphi_4 \\ -\mathrm{i}\,\partial_2\varphi_1 + 3\,\partial_3\varphi_1 + \mathrm{i}\,\partial_4\varphi_2 - \mathrm{i}\,\partial_1\varphi_4 \\ \mathrm{i}\,\partial_4\varphi_1 + \mathrm{i}\,\partial_2\varphi_2 + 3\,\partial_3\varphi_2 + \mathrm{i}\,\partial_1\varphi_3 \\ -\mathrm{i}\,\partial_1\varphi_2 - \mathrm{i}\,\partial_2\varphi_3 + 3\,\partial_3\varphi_3 + \mathrm{i}\,\partial_4\varphi_4) \\ \mathrm{i}\,\partial_1\varphi_1 + \mathrm{i}\,\partial_4\varphi_3 + \mathrm{i}(\partial_2 - 3\mathrm{i}\,\partial_3)\varphi_4) \\ \partial_4\varphi_1 - \partial_2\varphi_2 + \mathrm{i}\,\partial_3\varphi_2 - \partial_1\varphi_3 \\ \partial_2\varphi_1 + \mathrm{i}\,\partial_3\varphi_1 + \partial_4\varphi_2 + \partial_1\varphi_4 \\ -\partial_1\varphi_1 + \partial_4\varphi_3 - (\partial_2 - \mathrm{i}\,\partial_3)\varphi_4 \\ \partial_1\varphi_2 + \partial_2\varphi_3 + \mathrm{i}\,\partial_3\varphi_3 + \partial_4\varphi_4 \end{pmatrix} = 0$$

A.8) *SU (3) Gell-Mann λ-Matrices*

The structural constants of the eight-dimensional K-algebra of the λ matrices result from the relationship:

$$A_l^{jk} = \frac{1}{4\mathrm{i}} Trace([\lambda_{\mathrm{j}}, \lambda_{\mathrm{k}}]\,\lambda_{\mathrm{l}})$$

Here then by Eq. (7.21) two-handed P operators with the property exist: $P_l{:}\,\lambda \;\; = -P_r{:}\,\lambda$

$$P_r{:}\,\lambda =$$

$$\begin{pmatrix} 0 & \frac{\partial_3}{3} & -\frac{\partial_2}{3} & \frac{\partial_7}{6} & -\frac{\partial_6}{6} & \frac{\partial_5}{6} & -\frac{\partial_4}{6} & 0 \\ -\frac{\partial_3}{3} & 0 & \frac{\partial_1}{3} & \frac{\partial_6}{6} & \frac{\partial_7}{6} & -\frac{\partial_4}{6} & -\frac{\partial_5}{6} & 0 \\ \frac{\partial_2}{3} & -\frac{\partial_1}{3} & 0 & \frac{\partial_5}{6} & -\frac{\partial_4}{6} & -\frac{\partial_7}{6} & \frac{\partial_6}{6} & 0 \\ -\frac{\partial_7}{6} & -\frac{\partial_6}{6} & -\frac{\partial_5}{6} & 0 & \frac{\partial_3}{6} + \frac{\partial_8}{2\sqrt{3}} & \frac{\partial_2}{6} & \frac{\partial_1}{6} & -\frac{\partial_5}{2\sqrt{3}} \\ \frac{\partial_6}{6} & -\frac{\partial_7}{6} & \frac{\partial_4}{6} & -\frac{\partial_3}{6} - \frac{\partial_8}{2\sqrt{3}} & 0 & -\frac{\partial_1}{6} & \frac{\partial_2}{6} & \frac{\partial_4}{2\sqrt{3}} \\ -\frac{\partial_5}{6} & \frac{\partial_4}{6} & \frac{\partial_7}{6} & -\frac{\partial_2}{6} & \frac{\partial_1}{6} & 0 & \frac{\partial_8}{2\sqrt{3}} - \frac{\partial_3}{6} & -\frac{\partial_7}{2\sqrt{3}} \\ \frac{\partial_4}{6} & \frac{\partial_5}{6} & -\frac{\partial_6}{6} & -\frac{\partial_1}{6} & -\frac{\partial_2}{6} & \frac{\partial_3}{6} - \frac{\partial_8}{2\sqrt{3}} & 0 & \frac{\partial_6}{2\sqrt{3}} \\ 0 & 0 & 0 & \frac{\partial_5}{2\sqrt{3}} & -\frac{\partial_4}{2\sqrt{3}} & \frac{\partial_7}{2\sqrt{3}} & -\frac{\partial_6}{2\sqrt{3}} & 0 \end{pmatrix}$$

The generalized Cauchy-Riemann equations here are identical for both handed operators where the listed vector is to be set to zero:

$$
\begin{pmatrix}
\partial_1\varphi_1 \\
\frac{\partial_2\varphi_1}{3}+\frac{2\,\partial_1\varphi_2}{3}+\frac{\partial_3\varphi_4}{6}-\frac{\partial_4\varphi_5}{6}-\frac{\partial_7\varphi_6}{6}+\frac{\partial_6\varphi_7}{6} \\
\frac{\partial_3\varphi_1}{3}+\frac{2\,\partial_1\varphi_3}{3}-\frac{\partial_6\varphi_4}{6}-\frac{\partial_7\varphi_5}{6}+\frac{\partial_4\varphi_6}{6}+\frac{\partial_5\varphi_7}{6} \\
\frac{\partial_4\varphi_1}{12}+\frac{\partial_5\varphi_2}{12}-\frac{\partial_6\varphi_3}{12}+\frac{11\,\partial_1\varphi_4}{12}-\frac{\partial_2\varphi_5}{12}+\frac{\partial_3\varphi_6}{12}+(\frac{\partial_3}{12}-\frac{\partial_8}{4\sqrt{3}})\varphi_4+\frac{\partial_6\varphi_8}{4\sqrt{3}} \\
\frac{\partial_5\varphi_1}{12}-\frac{\partial_4\varphi_2}{12}-\frac{\partial_7\varphi_3}{12}+\frac{\partial_2\varphi_4}{12}+\frac{11\,\partial_1\varphi_5}{12}+(\frac{\partial_3}{12}-\frac{\partial_8}{4\sqrt{3}})\varphi_5+\frac{\partial_7\varphi_8}{4\sqrt{3}} \\
\frac{\partial_6\varphi_1}{12}-\frac{\partial_7\varphi_2}{12}+\frac{\partial_4\varphi_3}{12}+(-\frac{\partial_3}{12}-\frac{\partial_8}{4\sqrt{3}})\varphi_4+\frac{11\,\partial_1\varphi_6}{12}+\frac{\partial_2\varphi_7}{12}+\frac{\partial_4\varphi_8}{4\sqrt{3}} \\
\frac{\partial_7\varphi_1}{12}+\frac{\partial_6\varphi_2}{12}+\frac{\partial_5\varphi_3}{12}+(-\frac{\partial_3}{12}-\frac{\partial_8}{4\sqrt{3}})\varphi_5-\frac{\partial_2\varphi_6}{12}+\frac{11\,\partial_1\varphi_7}{12}+\frac{\partial_5\varphi_8}{4\sqrt{3}} \\
\partial_1\varphi_8 \\
\frac{2\,\partial_2\varphi_1}{3}+\frac{\partial_1\varphi_2}{3}-\frac{\partial_3\varphi_4}{6}+\frac{\partial_4\varphi_5}{6}+\frac{\partial_7\varphi_6}{6}-\frac{\partial_6\varphi_7}{6} \\
\partial_2\varphi_2 \\
\frac{\partial_3\varphi_2}{3}+\frac{2\,\partial_2\varphi_3}{3}+\frac{\partial_7\varphi_4}{6}-\frac{\partial_6\varphi_5}{6}+\frac{\partial_5\varphi_6}{6}-\frac{\partial_4\varphi_7}{6} \\
-\frac{1}{12}\partial_3\varphi_1+\frac{\partial_4\varphi_2}{12}+\frac{\partial_7\varphi_3}{12}+\frac{11\,\partial_2\varphi_4}{12}+\frac{\partial_1\varphi_5}{12}+(\frac{\partial_8}{4\sqrt{3}}-\frac{\partial_3}{12})\varphi_7-\frac{\partial_4\varphi_8}{4\sqrt{3}} \\
\frac{\partial_4\varphi_1}{12}+\frac{\partial_3\varphi_2}{12}-\frac{\partial_6\varphi_3}{12}-\frac{\partial_1\varphi_4}{12}+\frac{11\,\partial_2\varphi_5}{12}+(\frac{\partial_3}{4\sqrt{3}}-\frac{\partial_8}{12})\varphi_6+\frac{\partial_6\varphi_8}{4\sqrt{3}} \\
\frac{\partial_7\varphi_1}{12}+\frac{\partial_6\varphi_2}{12}+\frac{\partial_5\varphi_3}{12}+(-\frac{\partial_3}{12}-\frac{\partial_8}{4\sqrt{3}})\varphi_5+\frac{11\,\partial_2\varphi_6}{12}-\frac{\partial_1\varphi_7}{12}+\frac{\partial_5\varphi_8}{4\sqrt{3}} \\
-\frac{1}{12}\partial_4\varphi_1+\frac{\partial_7\varphi_2}{12}-\frac{\partial_4\varphi_3}{12}+(\frac{\partial_3}{12}+\frac{\partial_8}{4\sqrt{3}})\varphi_4+\frac{\partial_1\varphi_6}{12}+\frac{11\,\partial_2\varphi_7}{12}-\frac{\partial_4\varphi_8}{4\sqrt{3}} \\
\partial_2\varphi_8 \\
\frac{2\,\partial_3\varphi_1}{3}+\frac{\partial_1\varphi_3}{3}+\frac{\partial_4\varphi_4}{6}+\frac{\partial_7\varphi_5}{6}-\frac{\partial_4\varphi_6}{6}-\frac{\partial_5\varphi_7}{6} \\
\frac{2\,\partial_3\varphi_2}{3}+\frac{\partial_2\varphi_3}{3}-\frac{\partial_7\varphi_4}{6}+\frac{\partial_4\varphi_5}{6}-\frac{\partial_5\varphi_6}{6}+\frac{\partial_4\varphi_7}{6} \\
\partial_3\varphi_3 \\
\frac{\partial_4\varphi_1}{12}-\frac{\partial_1\varphi_2}{12}+\frac{\partial_4\varphi_3}{12}+(\frac{11\,\partial_3}{12}-\frac{\partial_8}{4\sqrt{3}})\varphi_4-\frac{\partial_1\varphi_5}{12}+\frac{\partial_2\varphi_7}{12}+\frac{\partial_4\varphi_8}{4\sqrt{3}} \\
\frac{\partial_5\varphi_1}{12}+\frac{\partial_4\varphi_2}{12}+\frac{\partial_5\varphi_3}{12}+(\frac{11\,\partial_3}{12}-\frac{\partial_8}{4\sqrt{3}})\varphi_5-\frac{\partial_2\varphi_6}{12}-\frac{\partial_1\varphi_7}{12}+\frac{\partial_5\varphi_8}{4\sqrt{3}} \\
-\frac{1}{12}\partial_4\varphi_1-\frac{\partial_4\varphi_2}{12}+\frac{\partial_4\varphi_3}{12}+\frac{\partial_1\varphi_4}{12}+\frac{\partial_2\varphi_5}{12}+(\frac{11\,\partial_3}{12}+\frac{\partial_8}{4\sqrt{3}})\varphi_6-\frac{\partial_4\varphi_8}{4\sqrt{3}} \\
-\frac{1}{12}\partial_5\varphi_1+\frac{\partial_4\varphi_2}{12}+\frac{\partial_7\varphi_3}{12}-\frac{\partial_2\varphi_4}{12}+\frac{\partial_1\varphi_5}{12}+(\frac{11\,\partial_3}{12}+\frac{\partial_8}{4\sqrt{3}})\varphi_7-\frac{\partial_5\varphi_8}{4\sqrt{3}} \\
\partial_4\varphi_8 \\
\frac{11\,\partial_4\varphi_1}{12}-\frac{\partial_5\varphi_2}{12}+\frac{\partial_4\varphi_3}{12}+\frac{\partial_1\varphi_4}{12}+\frac{\partial_2\varphi_5}{12}+(\frac{\partial_8}{4\sqrt{3}}-\frac{\partial_3}{12})\varphi_6-\frac{\partial_4\varphi_8}{4\sqrt{3}} \\
\frac{\partial_5\varphi_1}{12}+\frac{11\,\partial_4\varphi_2}{12}-\frac{\partial_7\varphi_3}{12}+\frac{\partial_2\varphi_4}{12}-\frac{\partial_1\varphi_5}{12}+(\frac{\partial_3}{12}-\frac{\partial_8}{4\sqrt{3}})\varphi_7+\frac{\partial_5\varphi_8}{4\sqrt{3}} \\
-\frac{1}{12}\partial_4\varphi_1+\frac{\partial_7\varphi_2}{12}+\frac{11\,\partial_4\varphi_3}{12}+(\frac{\partial_3}{12}+\frac{\partial_8}{4\sqrt{3}})\varphi_4+\frac{\partial_1\varphi_6}{12}-\frac{\partial_2\varphi_7}{12}-\frac{\partial_4\varphi_8}{4\sqrt{3}} \\
\partial_4\varphi_4 \\
\frac{\partial_2\varphi_1}{6}-\frac{\partial_1\varphi_2}{6}+\frac{\partial_3\varphi_4}{3}+\frac{2\,\partial_4\varphi_5}{3}+\frac{\partial_7\varphi_6}{6}-\frac{\partial_6\varphi_7}{6} \\
-\frac{1}{6}\partial_3\varphi_1+\frac{\partial_4\varphi_3}{6}+\frac{\partial_1\varphi_4}{6}+\frac{\partial_7\varphi_5}{12}+\frac{11\,\partial_4\varphi_6}{12}-\frac{\partial_5\varphi_7}{12} \\
\frac{\partial_3\varphi_2}{6}-\frac{\partial_2\varphi_3}{6}+\frac{\partial_7\varphi_4}{6}-\frac{\partial_6\varphi_5}{12}+\frac{\partial_5\varphi_6}{12}+\frac{11\,\partial_4\varphi_7}{12} \\
-\frac{\partial_4\varphi_1}{4\sqrt{3}}+\frac{\partial_7\varphi_2}{4\sqrt{3}}-\frac{\partial_6\varphi_3}{4\sqrt{3}}+(\frac{\partial_3}{4\sqrt{3}}+\frac{\partial_8}{4})\varphi_4+\frac{\partial_5\varphi_6}{4\sqrt{3}}-\frac{\partial_2\varphi_7}{4\sqrt{3}}+\frac{3\,\partial_4\varphi_8}{4} \\
\frac{11\,\partial_5\varphi_1}{12}+\frac{\partial_4\varphi_2}{12}+\frac{\partial_7\varphi_3}{12}-\frac{\partial_2\varphi_4}{12}+\frac{\partial_1\varphi_5}{12}+(\frac{\partial_8}{4\sqrt{3}}-\frac{\partial_3}{12})\varphi_7-\frac{\partial_1\varphi_8}{4\sqrt{3}} \\
-\frac{1}{12}\partial_4\varphi_1+\frac{11\,\partial_5\varphi_2}{12}+\frac{\partial_6\varphi_3}{12}+\frac{\partial_1\varphi_4}{12}+\frac{\partial_2\varphi_5}{12}+(\frac{\partial_8}{4\sqrt{3}}-\frac{\partial_3}{12})\varphi_6-\frac{\partial_4\varphi_8}{4\sqrt{3}} \\
-\frac{1}{12}\partial_7\varphi_1-\frac{\partial_6\varphi_2}{12}+\frac{11\,\partial_5\varphi_3}{12}+(\frac{\partial_3}{12}+\frac{\partial_8}{4\sqrt{3}})\varphi_5+\frac{\partial_2\varphi_6}{12}+\frac{\partial_1\varphi_7}{12}-\frac{\partial_5\varphi_8}{4\sqrt{3}} \\
\partial_5\varphi_5 \\
-\frac{1}{6}\partial_2\varphi_1+\frac{\partial_1\varphi_2}{6}+\frac{2\,\partial_5\varphi_4}{3}+\frac{\partial_4\varphi_5}{3}-\frac{\partial_7\varphi_6}{6}+\frac{\partial_6\varphi_7}{6} \\
-\frac{1}{6}\partial_3\varphi_2+\frac{\partial_2\varphi_3}{6}-\frac{\partial_7\varphi_4}{12}+\frac{\partial_6\varphi_5}{12}+\frac{11\,\partial_5\varphi_6}{12}+\frac{\partial_4\varphi_7}{12} \\
-\frac{1}{6}\partial_3\varphi_1+\frac{\partial_1\varphi_3}{6}+\frac{\partial_6\varphi_4}{12}+\frac{\partial_7\varphi_5}{12}-\frac{\partial_4\varphi_6}{12}+\frac{11\,\partial_5\varphi_7}{12} \\
-\frac{\partial_7\varphi_1}{4\sqrt{3}}-\frac{\partial_6\varphi_2}{4\sqrt{3}}-\frac{\partial_5\varphi_3}{4\sqrt{3}}+(\frac{\partial_3}{4\sqrt{3}}+\frac{\partial_8}{4})\varphi_5+\frac{\partial_2\varphi_6}{4\sqrt{3}}+\frac{\partial_1\varphi_7}{4\sqrt{3}}+\frac{3\,\partial_5\varphi_8}{4} \\
\frac{11\,\partial_6\varphi_1}{12}+\frac{\partial_7\varphi_2}{12}-\frac{\partial_4\varphi_3}{12}+(\frac{\partial_3}{12}+\frac{\partial_8}{4\sqrt{3}})\varphi_4+\frac{\partial_1\varphi_6}{12}-\frac{\partial_2\varphi_7}{12}-\frac{\partial_4\varphi_8}{4\sqrt{3}} \\
-\frac{1}{12}\partial_7\varphi_1+\frac{11\,\partial_6\varphi_2}{12}-\frac{\partial_1\varphi_3}{12}-\frac{\partial_2\varphi_4}{12}+(\frac{\partial_3}{12}+\frac{\partial_8}{4\sqrt{3}})\varphi_5+\frac{\partial_2\varphi_6}{12}+\frac{\partial_1\varphi_7}{12}-\frac{\partial_5\varphi_8}{4\sqrt{3}} \\
\frac{\partial_4\varphi_1}{12}+\frac{\partial_5\varphi_2}{12}+\frac{11\,\partial_6\varphi_3}{12}-\frac{\partial_1\varphi_4}{12}-\frac{\partial_2\varphi_5}{12}+(\frac{\partial_3}{12}-\frac{\partial_8}{4\sqrt{3}})\varphi_6+\frac{\partial_4\varphi_8}{4\sqrt{3}} \\
\frac{\partial_1\varphi_1}{6}-\frac{\partial_1\varphi_3}{6}+\frac{11\,\partial_2\varphi_4}{12}-\frac{\partial_7\varphi_5}{12}+\frac{\partial_4\varphi_6}{12}+\frac{\partial_5\varphi_7}{12} \\
\frac{\partial_2\varphi_2}{6}-\frac{\partial_2\varphi_3}{6}+\frac{\partial_7\varphi_4}{12}+\frac{11\,\partial_6\varphi_5}{12}+\frac{\partial_5\varphi_6}{12}-\frac{\partial_4\varphi_7}{12} \\
\partial_6\varphi_6 \\
-\frac{1}{6}\partial_1\varphi_1+\frac{\partial_1\varphi_2}{6}+\frac{\partial_2\varphi_4}{6}-\frac{\partial_4\varphi_5}{6}+\frac{\partial_7\varphi_6}{3}+\frac{2\,\partial_6\varphi_7}{3} \\
-\frac{\partial_4\varphi_1}{4\sqrt{3}}-\frac{\partial_5\varphi_2}{4\sqrt{3}}+\frac{\partial_1\varphi_4}{4\sqrt{3}}+\frac{\partial_2\varphi_5}{4\sqrt{3}}+(\frac{\partial_8}{4}-\frac{\partial_3}{4\sqrt{3}})\varphi_6+\frac{3\,\partial_6\varphi_8}{4} \\
\frac{11\,\partial_7\varphi_1}{12}-\frac{\partial_4\varphi_2}{12}-\frac{\partial_5\varphi_3}{12}+(\frac{\partial_3}{12}+\frac{\partial_8}{4\sqrt{3}})\varphi_5+\frac{\partial_2\varphi_6}{12}+\frac{\partial_1\varphi_7}{12}-\frac{\partial_5\varphi_8}{4\sqrt{3}} \\
\frac{\partial_6\varphi_1}{12}+\frac{11\,\partial_7\varphi_2}{12}+\frac{\partial_1\varphi_3}{12}+(-\frac{\partial_3}{12}-\frac{\partial_8}{4\sqrt{3}})\varphi_4-\frac{\partial_1\varphi_6}{12}+\frac{\partial_2\varphi_7}{12}+\frac{\partial_4\varphi_8}{4\sqrt{3}} \\
\frac{\partial_5\varphi_1}{12}-\frac{\partial_4\varphi_2}{12}+\frac{11\,\partial_7\varphi_3}{12}+\frac{\partial_2\varphi_4}{12}-\frac{\partial_1\varphi_5}{12}+(\frac{\partial_3}{12}-\frac{\partial_8}{4\sqrt{3}})\varphi_7+\frac{\partial_5\varphi_8}{4\sqrt{3}} \\
-\frac{1}{6}\partial_2\varphi_2+\frac{\partial_2\varphi_3}{6}+\frac{11\,\partial_7\varphi_4}{12}+\frac{\partial_4\varphi_5}{12}-\frac{\partial_5\varphi_6}{12}+\frac{\partial_4\varphi_7}{12} \\
\frac{\partial_1\varphi_1}{6}-\frac{\partial_1\varphi_3}{6}-\frac{\partial_2\varphi_4}{12}+\frac{11\,\partial_7\varphi_5}{12}+\frac{\partial_4\varphi_6}{12}+\frac{\partial_5\varphi_7}{12} \\
\frac{\partial_2\varphi_1}{6}-\frac{\partial_2\varphi_3}{6}-\frac{\partial_7\varphi_4}{6}+\frac{\partial_4\varphi_5}{6}+\frac{2\,\partial_7\varphi_6}{3}+\frac{\partial_6\varphi_7}{3} \\
\partial_7\varphi_7 \\
-\frac{\partial_5\varphi_1}{4\sqrt{3}}+\frac{\partial_4\varphi_2}{4\sqrt{3}}+\frac{\partial_7\varphi_3}{4\sqrt{3}}-\frac{\partial_2\varphi_4}{4\sqrt{3}}+\frac{\partial_1\varphi_5}{4\sqrt{3}}+(\frac{\partial_8}{4}-\frac{\partial_3}{4\sqrt{3}})\varphi_7+\frac{3\,\partial_7\varphi_8}{4} \\
\partial_8\varphi_1 \\
\partial_8\varphi_2 \\
\partial_8\varphi_3 \\
\frac{\partial_6\varphi_1}{4\sqrt{3}}-\frac{\partial_7\varphi_2}{4\sqrt{3}}+\frac{\partial_4\varphi_3}{4\sqrt{3}}+(\frac{3\,\partial_8}{4}-\frac{\partial_3}{4\sqrt{3}})\varphi_4-\frac{\partial_1\varphi_6}{4\sqrt{3}}+\frac{\partial_2\varphi_7}{4\sqrt{3}}+\frac{\partial_4\varphi_8}{4} \\
\frac{\partial_7\varphi_1}{4\sqrt{3}}+\frac{\partial_6\varphi_2}{4\sqrt{3}}+\frac{\partial_5\varphi_3}{4\sqrt{3}}+(\frac{3\,\partial_8}{4}-\frac{\partial_3}{4\sqrt{3}})\varphi_5-\frac{\partial_2\varphi_6}{4\sqrt{3}}-\frac{\partial_1\varphi_7}{4\sqrt{3}}+\frac{\partial_5\varphi_8}{4} \\
\frac{\partial_4\varphi_1}{4\sqrt{3}}+\frac{\partial_5\varphi_2}{4\sqrt{3}}-\frac{\partial_4\varphi_3}{4\sqrt{3}}-\frac{\partial_1\varphi_4}{4\sqrt{3}}-\frac{\partial_2\varphi_5}{4\sqrt{3}}+(\frac{\partial_3}{4\sqrt{3}}+\frac{3\,\partial_8}{4})\varphi_6+\frac{\partial_4\varphi_8}{4} \\
\frac{\partial_5\varphi_1}{4\sqrt{3}}-\frac{\partial_4\varphi_2}{4\sqrt{3}}-\frac{\partial_7\varphi_3}{4\sqrt{3}}+\frac{\partial_4\varphi_4}{4\sqrt{3}}-\frac{\partial_1\varphi_5}{4\sqrt{3}}+(\frac{\partial_3}{4\sqrt{3}}+\frac{3\,\partial_8}{4})\varphi_7+\frac{\partial_7\varphi_8}{4} \\
\partial_8\varphi_8
\end{pmatrix}
= 0
$$

References

[1] Rechnender Raum, K.Zuse, *Elektronische Datenverarbeitung*, Bd. 8, pp 336, 1967

[2] Geometrie und Erfahrung, A.Einstein, *Preussische Akadamie der Wissenschaften,* Berlin*,* 1921

[3] Conference of Physics of Computation, USA Boston 1981 May 6-8

[4] The Theory of Positrons, R.Feynman, *Physical Review* 76, pp 749, 1949

[5] Logik der Forschung, *Buchreihe Wiener Kreis*, K.Popper, 1934

[6] The Quantum Theory of the Electron, P.Dirac, *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 1928

[7] Zur Quantenmechanik des magnetischen Elektrons, W.Pauli, *Zeitschrift für Physik*, Bd. 43, S. 601, 1927

[8] Symmetries of baryons and mesons, M.Gell-Mann, *Physical Review 125 (3) 1067,* 1962

[9] A Dynamical Theory of the Electromagnetic Field, J.C. Maxwell, *Royal Society Transactions*, 155, pp 459–512, 1865

[10] Die funktionentheoretischen Beziehungen der Maxwell Aethergleichungen, Dissertation, K.Lánczos, Technische Hochschule Budapest, 1919

[11] Über den Zusammenhang zwischen den Cauchy-Riemannschen und Diracschen Differentialgleichungen, D.Iwanenko und K.Solsky, *K. Z. Physik 63*: 129, Universität Leningrad und Charkow, 1930

[12] Die Grundlage der allgemeinen Relativitätstheorie, A.Einstein, *Annalen der Physik*, 4.Folge Bd. 44, pp50, 1916

[13] Sur la dynamique de l'électron, H.Poincaré, Henri, *Rendiconti del Circolo matematico di Palermo,* 21, S. 129–176,1906

[14] On the reciprocal of the general algebraic matrix, E.H. Moore, *Bulletin of the American Mathematical Society*, 26, 395-395, 1920 / A generalized inverse for matrices, R.Penrose, *Proceedings of the Cambridge Philosophical society*, 51, 406-413, 1955

[15] Some theorems on matrix differentiation with special reference to Kronecker matrix products, H.Neudecker, *Journal American Statistics Assoc*., Volume 64 pp 953-963, 1969

[16] Some consequences of a theorem of Bott, J.Milnor, *Annals of Mathematics*, Band 68, S. 444–449, 1958

Open source implementation of Algebraic Field Theory

The "King's path" followed up here, is to have a complete generator free formalism for any K-algebra at the end, avoiding all issues of handling non-commutative or even non-associative generators. Any K-algebra (https://en.wikipedia.org/wiki/Algebra_over_a_field) which is constructed from a finite base of generators (https://en.wikipedia.org/wiki/Generator_(mathematics)) is defined by a related (2,1) tensor of the structure coefficients. The generator free formalism is realized by using rectangular matrix calculus in combination with the Kronecker product (symbol $\otimes$) and triple index permutation operators, acting onto a rectangular matrix representation of the structure coefficients tensor. To move forward from complex analyis to algebraic analyis, Algebraic Field Theory is using minimal Dirac operators as hyper complex derivation. This is a different approach, motivated from quantum physics, as mimicking the complex numbers for other K-algebras by usual procedures. However, in the special case of the complex numbers leading to the same results. All methods of this formalism are collected under a Python class *K_Algebra()*. This script is done in form of a Jupyter notebook with SageMath kernel. Jupyter, SageMath, Python are all free open source tools and free of charge. Goal is to demonstrate the feasibility and how Algebraic Field Theory works in practise.

Examples for K-algebras are the complex numbers, the vector product, quaternions, octonions, all quantum algebras and much more but also even logical operations like the AND/OR can be understood as a K-algebra of a binary operation.

Robert H. Ihde (roberthans.ihde@gmail.com), Munich, Germany

revision A, August 2019: Jupyter Notebook file=K-algebra_revA.ipynb, status draft, subject to changes and error corrections

Open source tools used: SageMath (http://www.sagemath.org) 8.8, Jupyter (https://jupyter.org) notebook 5.7.6, Python (http://www.python.org) 2.7.15

Following key features of the class *K_Algebra()* will be provided:

- create a number *q* of the instance (implicit as a given variable or explicite as field number)
- multiply two numbers *p,q* of the instance (as method or in infix notation $p \otimes q$)
- check for commutativity and associativity
- determine the identity (the algebraic one) and chirality (right, left), if available under certain constraints
- determine the dual space operators of the generators, if available under certain constraints
- determine the dual space operators of the field (projected metric), if available under certain constraints
- mapping cartesian products of K-algebras to a higher dim K-algebra over the reals
- determine the minimal Dirac operator for any K-algebra
- determine the minimal Cauchy-Riemann equations for any K-algebra
- a predefined dictionary of multiplication tables for usual K-algebras

---

1. Introduction

A K-algebra $\mathbb{A}$ which is created from a finite n-dimensional generator base can be defined by the related structure coefficients (https://en.wikipedia.org/wiki/Structure_constants) of a triple indexed tensor.
This (2,1) tensor will be mapped to a corresponding (n^2, n) rectangular matrix ***A*** over the field, symbolically $\mathbb{A} \rightarrow \boldsymbol{A}$.

Any finite n-dimensional binary product can be re-written as a vectorized version of the underlying multiplication table of the related K-algebra over the field $\mathbb{K}$, using the Kronecker product $\otimes$ and a column vector representation *g* of the generator elements. Summarizing, the rectangular matrix ***A*** is conveying fully the multiplication rules of the generators, defining a map of the K-algebra $\mathbb{A}$ over the field $\mathbb{K}$ to the related rectangular matrix ***A***.

$g \otimes_{\mathbb{A}} g := \boldsymbol{A} \cdot g$

In tensor index notation, where a_l^{jk} being the structure coefficients (weights) of ***A***:

$g^j *_{\mathbb{A}} g^k := \sum_{l=1}^{n} a_l^{jk} g^l$ with generators $g^j \notin \mathbb{K}$ and $a_l^{jk} \in \mathbb{K}$

The triple indexed structure coefficients are mapped to a corresponding rectangular matrix ***A*** with the help of unit vectors e_j as follows, using the superscript ***T*** operator as symbol for the transpose() function:

$e^j = e_j^T$ and $e^j e_k = \delta_k^j$

$\boldsymbol{A} = \sum_{j,k,l=1}^{n} a_l^{jk} (e_j \otimes e_k \otimes e^l)$

A generic element of a related K-algebra, labeled $\tilde{q}$, is constructed by using the scalar product of the generators and a corresponding field vector *q* of same dimension. This is allowed because the generators and the field do commute by definition in the classical representation.

$\tilde{q} := g^T q = \sum_{j=1}^{n} g^j q_j = q^T g$

Exemplary for a complex number $\tilde{q} = 2 \cdot 1 + 3 \cdot i = 1 \cdot 2 + i \cdot 3$ "classical representation of a complex number",
compared to the "generator free representation" as vector $\Rightarrow g = \begin{pmatrix} 1 \\ i \end{pmatrix}, q = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$.

Summarizing, small letters with a tilde as hat are classic representations of algebraic numbers (complex, quaternions etc.) and pure small letters are vectors over the field, representing the generator free representation of algebraic numbers.

In this example as in available literatures about complex number systems, the complex one with the real one is often identified on the basis of the same algebraic properties ($g_1 \equiv 1$). Via the strict mathematical view, all generators are always of a different kind than the field numbers which is especially valid for K-algebras without identity. In the generator free approach, these kind of problems are avoided.

Building the binary product between two elements $\tilde{p}$ and $\tilde{q}$, using the Kroneckerproduct:

$\tilde{p} \otimes \tilde{q} = (g^T p) \otimes (g^T q)$

and applying now the Kronecker identity $(ab) \otimes (cd) = (a \otimes c)\,(b \otimes d)$, with $\tilde{r}$ for the resulting number:

$(g^T p) \otimes (g^T q) = (g^T \otimes g^T)\,(p \otimes q) = g^T\ \boldsymbol{A}^T\,(p \otimes q) = g^T r$

Any binary multiplication of a K-algebra, constructed from a finite generator basis, can be therefore mapped to a generator free representation by:

$r = \boldsymbol{A}^T\,(p \otimes q)$ in index notation $r_j = \sum_{k,l=1}^{n} a_j^{kl} p_k q_l$

This helps very much in computer calculations of any K-algebras because only rectangular matrices, vectors over the field are required. The handling of non-commutative or even non-associative generators in coding is challenging because the sequence of the generators must be carefully taken into account in all related algebraic expressions. With the above rectangular matrix/Kronecker product rule, any K-algebra can be mapped to the matrix/vector domain.
The generators are then only needed to display at the end the classical representation of algebraic numbers or to display the rectangular matrix ***A*** (unique to a K-algebra $\mathbb{A}$) in form of a (n, n) quadratic multiplication table of the generators.

How to determine the specific structure coefficients values (reffered also as weights) of the rectangular matrix ***A*** which represents a given K-algebra?

The following manual steps are showing one possible procedure.
Later a method will be provided which can directly create the rectangular matrix from a given formatted multiplication table.

Let's start with the complex numbers as example to see the correspondencies.

Because the multiplication table for the complex numbers consists only of integer coefficients, first a module over the integers is defined.

In [1]:
```
%display latex # for a traditional symbolic math output in the Sage Jupyter notebook
```

In [2]:
```
Z2 = FiniteRankFreeModule(ZZ,2,name='Z2', start_index=1); print(Z2)
```

```
Rank-2 free module Z2 over the Integer Ring
```

Now the (2,1) tensor for the structure coefficients can be defined over the integer ring.

In [3]:
```
K2 = Z2.tensor((2,1),name='K2'); print(K2)
```

```
Type-(2,1) tensor K2 on the Rank-2 free module Z2 over the Integer Ring
```

To assign values for this tensor first a symbolical, arbitrary basis for the tensor is required.

In [4]:
```
b = Z2.basis('b'); b
```

Out[4]: (b_1, b_2)

The multiplication table for the complex numbers is (with 'e' being here the generator symbol for the complex identity):

$e * e \to 1 \cdot e,\ e * i \to 1 \cdot i,$

$i * e \to 1 \cdot i,\ i * i \to -1 \cdot e$

Therefore the following weights K2[] as projection from the generators to the field of the (2,1) tensor must be set accordingly.

The procedure, exemplary for the index [2,1,2], is so that the first two indices [(2,1),()] are representing the binary product between here the 2nd generator = i and the first generator = e in the exact sequence. The third index [(),(2)] is the related generator on the result side, here the 2nd generator = i. The concrete field value of the weight for this generator, here $1 \cdot i$ is "1" and this is the searched value for K2[2,1,2]. What is now the value for K2[2,1,1]? The complex numbers do not have a projection from $i * e \to e$, therefore $i * e = 0 \cdot e$ and K2[2,1,1]=0.

Conveniently this tensor definition already initializes all weights already to 0, only the non-zero entries must be defined.

In [5]:
```
K2[1,1,1] = 1; K2[1,2,2] = 1;
K2[2,1,2] = 1; K2[2,2,1] = -1;
```

The tensor of the structure coefficients, representing the complex numbers is then:

In [6]:
```
K2[:]
```

Out[6]: $[[[1, 0], [0, 1]], [[0, 1], [-1, 0]]]$

The difference between a (2,1) tensor and a mapped related rectangular matrix is now at hand.
The (2,1) tensor can be geometrically interpreted as a 3-dim cube of the structure coefficients whether the rectangular matrix is a projection to the 2-dim field domain. The difference here between a (2,1) tensor and a related (n^2,n) rectangular matrix is just the structure.
To get to the related rectangular matrix $\boldsymbol{A}_{\mathbb{C}}$, representing the K-algebra of the complex numbers, the tensor structure must be flattened as follows:

In [7]:
```
A_C = matrix(flatten(K2[:],max_level = 1)); A_C
```

Out[7]:
$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ -1 & 0 \end{pmatrix}$$

Let's check this out by first defining two arbitrary column vectors over a symbolic ring (p, q):

In [8]:
```
(p_1,p_2)= var('p_1','p_2'); p = matrix(SR,[p_1,p_2]).transpose();
(q_1,q_2)= var('q_1','q_2'); q = matrix(SR,[q_1,q_2]).transpose();
p,q
```

Out[8]:
$$\left(\begin{pmatrix} p_1 \\ p_2 \end{pmatrix}, \begin{pmatrix} q_1 \\ q_2 \end{pmatrix}\right)$$

Create the algebraic product $r = \boldsymbol{A}_{\mathbb{C}}^{T}(p \otimes q)$:

In [9]:
```
r=A_C.transpose()*(p.tensor_product(q));r
```

Out[9]:
$$\begin{pmatrix} p_1 q_1 - p_2 q_2 \\ p_2 q_1 + p_1 q_2 \end{pmatrix}$$

Move to the classic representation $\tilde{r}$ by using the generator base (1,i):

In [10]:
```
matrix([1,I])*r
```

Out[10]:
$$\left(p_1 q_1 + i\, p_2 q_1 + i\, p_1 q_2 - p_2 q_2\right)$$

Compared to Sage implementation of complex numbers:

In [11]:
```
expand((matrix([1,I])*p) * (matrix([1,I])*q))
```

Out[11]:
$$\left(p_1 q_1 + i\, p_2 q_1 + i\, p_1 q_2 - p_2 q_2\right)$$

Summarizing, the generator free multiplication of eq. [9] is already sufficient to map any K-algebra of a binary product!
A generalized algebraic number is represented by a related n-dimensional column vector.
The following chapters will demonstrate the capabilities of the class *K_Algebra()* first.
In the appendix, the mathematical derivation behind the methods is documented.

---

2. Demonstration of the methods

2.1 Initializing a K-algebra

By installing the open source tool SageMath already the installations for the open source tools Jupyter and Python are done in parallel. Run SageMath via the icon SageMath Notebook or via the command "$ sage --notebook jupyter" from the sage shell. This will start the Jupyter file explorer in the home directory. Put the library file K_Algebra.sage (Python source appended) to this home directory. Alternatively you need to specify the concrete path to this library file for loading. To initialize the class methods just load the library. This is all and have fun by playing around with K-algebras and doing Algebraic Field Theory ☺.

In [12]:
```
load("K_Algebra.sage");
```

First setup an arbitrary instance of a K-algebra for a binary operation of finite dimension n and with denominator d for the structure values.
By default the variable identifier is set to 'a' for the structure coefficients of the K-algebra.
This can be changed (for example change to 's') by K_algebra(n,'s'). As example, here a new instance A2 for a two-dimensional K-algebra is defined by:

In [13]:
```
A2 = K_Algebra(2)
```

Method: .n
returns the dimension of the initialized K-algebra.

In [14]:
```
A2.n
```

Out[14]: 2

Method: .ini
is the automatic, arbitrary rectangular matrix with the set identifier (here = 'a') of dim = n.
The .ini value is a constant and will not be changed.
This is needed to return from a specific K-algebra to the arbitrary one or for comparison.

In [15]:
```
A2.ini
```

Out[15]:
$$\begin{pmatrix} a_{111} & a_{112} \\ a_{121} & a_{122} \\ a_{211} & a_{212} \\ a_{221} & a_{222} \end{pmatrix}$$

Method: .A
get a representation for the active rectangular matrix A (default = .ini for the dimension).
This variable can be overwritten by the set_algebra() method.

In [16]:
```
A2.A
```

Out[16]:
$$\begin{pmatrix} a_{111} & a_{112} \\ a_{121} & a_{122} \\ a_{211} & a_{212} \\ a_{221} & a_{222} \end{pmatrix}$$

Method: .set_algebra(A)
is setting the active K-algebra to a specific A of same dimension.

In the introduction the rectangular matrix for the complex numbers $\boldsymbol{A}_{\mathbb{C}}$ has been derived.
This is taken now as input to set the active K-algebra to the complex numbers.

In [17]:
```
A2.set_algebra(A_C)
```

In [18]:
```
A2.A, A2.ini  #compare the active K-algebra with the generic one
```

Out[18]:
$$\left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ -1 & 0 \end{pmatrix}, \begin{pmatrix} a_{111} & a_{112} \\ a_{121} & a_{122} \\ a_{211} & a_{212} \\ a_{221} & a_{222} \end{pmatrix}\right)$$

Method: .number(d)
create a number of the K-algebra with identifier d.
In the generator free representation the number is a column matrix vector.
If the identifier is a list of numbers = [2,3] then an explicit number is returned.

In [19]:
```
A2.number('q')
```

Out[19]:
$$\begin{pmatrix} q_1 \\ q_2 \end{pmatrix}$$

In [20]:
```
A2.number([2,3])
```

Out[20]:
$$\begin{pmatrix} 2 \\ 3 \end{pmatrix}$$

Method: .multiply(p,q)
multiplies two numbers algebraically with the active K-algebra. In the generator free version the result is also a vector.
Alternatively the infix notation p *A* q is possible.

In [21]:
```
p=A2.number('p'); q=A2.number('q');
p,q,A2.multiply(p,q)
```

Out[21]:
$$\left(\begin{pmatrix} p_1 \\ p_2 \end{pmatrix}, \begin{pmatrix} q_1 \\ q_2 \end{pmatrix}, \begin{pmatrix} p_1 q_1 - p_2 q_2 \\ p_2 q_1 + p_1 q_2 \end{pmatrix}\right)$$

In [22]:
```
p *A* q                                  # first coordinate is the real part, second the imaginary part
```

Out[22]:
$$\begin{pmatrix} p_1 q_1 - p_2 q_2 \\ p_2 q_1 + p_1 q_2 \end{pmatrix}$$

In [23]:
```
A2.number([1,2]) *A* A2.number([-3,4])    # two explicit numbers (1+2i)*(-3+4i)
```

Out[23]:
$$\begin{pmatrix} -11 \\ -2 \end{pmatrix}$$

Method: .multi_tab()
outputs the multiplication table of the active K-algebra.
The multiplication table is using the indexed identifier g as symbol for the generator base.
The output is a n x n quadratic matrix without header row and header column.
Self defined multiplication tables are requiring the same format and indexed identifier g.
The table is just symbollically to be understood.

Be aware of:
In any case this table should not be used for further calculations.
The generators are here standard variables which will commute by further processing!
The table is just created in matrix format for output purposes and has only a symbolical meaning.

In [24]:
```
A2.multi_tab()  # the multiplication table for the complex numbers
```

Out[24]:
$$\begin{pmatrix} g_1 & g_2 \\ g_2 & -g_1 \end{pmatrix}$$

Let's go now the other way, there is a multiplication table available and the representative, rectangular matrix is searched for.
For this purpose we will use the **split-complex numbers (https://en.wikipedia.org/wiki/Split-complex_number)** where $i^2 = +1$.

The related multiplication is already available in the dictionary Dict_M2A{} by the identifier "SC".

In [25]:
```
M_SC = Dict_M2A.get("SC"); M_SC
```

Out[25]:
$$\begin{pmatrix} g_1 & g_2 \\ g_2 & g_1 \end{pmatrix}$$

Method: .A_from_multi_tab(M)
creates and automatically loads the related K-algebra from a multiplication table.
For the multiplication table the indexed identifier g as symbol for the indexed generator base has to be used.
The input is a n x n quadratic matrix without header row and header column.
The related rectangular matrix is returned and already automatically loaded by the method .A as active.

In [26]:
```
A_SC = A2.A_from_multi_tab(M_SC); A_SC
```

Out[26]:
$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$$

In [27]:
```
p *A* q   # the split complex product in the generator free view
```

Out[27]:
$$\begin{pmatrix} p_1 q_1 + p_2 q_2 \\ p_2 q_1 + p_1 q_2 \end{pmatrix}$$

Let's reset the active K-algebra again to the original arbitrary K-algebra.
This now ends this chapter by showing the arbitrary product and arbitrary multiplication table for any 2-dim K-algebra.

In [28]:
```
A2.set_algebra(A2.ini)
```

In [29]:
```
p *A* q
```

Out[29]:
$$\begin{pmatrix} a_{111} p_1 q_1 + a_{211} p_2 q_1 + a_{121} p_1 q_2 + a_{221} p_2 q_2 \\ a_{112} p_1 q_1 + a_{212} p_2 q_1 + a_{122} p_1 q_2 + a_{222} p_2 q_2 \end{pmatrix}$$

In [30]:
```
A2.multi_tab()
```

Out[30]:
$$\begin{pmatrix} a_{111} g_1 + a_{112} g_2 & a_{121} g_1 + a_{122} g_2 \\ a_{211} g_1 + a_{212} g_2 & a_{221} g_1 + a_{222} g_2 \end{pmatrix}$$

2.2 Higher functionalities

2.2.1 Check for commutativity and associativity

> **Method:** .is_commutative()
> tests if the K-algebra is commutative [p,q] == 0.
> The test will return literally:
> * commutative , if test is true
> * anti-commutative, if {p,q}==0
> * not commutative, if test is false

Let's use the 3-dim vector cross product, available already in the dictionary by the key="VP3".
For this purpose first the instance of a 3-dim K-algebra is required.

In [31]:
```
A3 = K_Algebra(3)
```

The result is the Levi-Civita ϵ tensor, represented as rectangular matrix:

In [32]:
```
A_VP3=A3.A_from_multi_tab(Dict_M2A.get("VP3")); A_VP3,A3.ini
```

Out[32]:
$$\left(\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} a_{111} & a_{112} & a_{113} \\ a_{121} & a_{122} & a_{123} \\ a_{131} & a_{132} & a_{133} \\ a_{211} & a_{212} & a_{213} \\ a_{221} & a_{222} & a_{223} \\ a_{231} & a_{232} & a_{233} \\ a_{311} & a_{312} & a_{313} \\ a_{321} & a_{322} & a_{323} \\ a_{331} & a_{332} & a_{333} \end{pmatrix}\right)$$

In [33]:
```
A3.is_commutative()       # the vector product is anti-commutative
```

Out[33]:
```
anti-commutative
```

In [34]:
```
p = A3.number('p'); q=A3.number('q');
p *A* q + q *A* p
```

Out[34]:
$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

In [35]:
```
p *A* q   # vector cross product in the generator free view
```

Out[35]:
$$\begin{pmatrix} -p_3q_2 + p_2q_3 \\ p_3q_1 - p_1q_3 \\ -p_2q_1 + p_1q_2 \end{pmatrix}$$

In [36]:
```
vector(p).cross_product(vector(q))  # SageMath implementation for comparison
```

Out[36]:
$$(-p_3q_2 + p_2q_3,\ p_3q_1 - p_1q_3,\ -p_2q_1 + p_1q_2)$$

> **Method:** .is_associative()
> tests if the K-algebra is associative (pq)r-p(qr) == 0,
> The test will return literally:
> * associative , if test is true
> * not associative, if test is false

In [37]:
```
A3.is_associative()  # the vector product is not associative
```

Out[37]:
```
not associative
```

In [38]:
```
r = A3.number('r');
(p *A* q) *A* r - p *A* (q *A* r)
```

Out[38]:
$$\begin{pmatrix} (q_2r_1 - q_1r_2)p_2 + (q_3r_1 - q_1r_3)p_3 + (p_2q_1 - p_1q_2)r_2 + (p_3q_1 - p_1q_3)r_3 \\ -(q_2r_1 - q_1r_2)p_1 + (q_3r_2 - q_2r_3)p_3 - (p_2q_1 - p_1q_2)r_1 + (p_3q_2 - p_2q_3)r_3 \\ -(q_3r_1 - q_1r_3)p_1 - (q_3r_2 - q_2r_3)p_2 - (p_3q_1 - p_1q_3)r_1 - (p_3q_2 - p_2q_3)r_2 \end{pmatrix}$$

Be aware of:
For more than two multiplicands the correct bracketing is essential!
The system interprets p*q*r without bracketing always as (p*q)*r.

In [39]:
```
(p *A* q) *A* r == p *A* q *A* r
```

Out[39]: True

2.2.2 Determine the identity (algebraic one) and chirality

Method: .one()
returns the identity, if exists.
The method will return:
* the generator free vector representation of the identity, if exists
* {} if no identity exists

The active vector product can't have any identity already from the geometric prospective.

In [40]:
```
A3.one()
```

Out[40]: {}

Moving back to the complex numbers (loaded here from the dictionary with index = "C"), is giving the expected result in the generator free view.

In [41]:
```
A2.A_from_multi_tab(Dict_M2A.get("C")); A2.A, A2.ini
```

Out[41]:
$$\left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ -1 & 0 \end{pmatrix}, \begin{pmatrix} a_{111} & a_{112} \\ a_{121} & a_{122} \\ a_{211} & a_{212} \\ a_{221} & a_{222} \end{pmatrix}\right)$$

In [42]:
```
A2.one()    # the identity of the complex numbers
```

Out[42]:
$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

Method: .type_of_one()
returns the chirality, if the identity exists.
The method will return:
* left-handed, if there exists only a left one
* right-handed, if there exists only a right one
* both-handed, if the identity can act as left and right one
* {} if no identity exists

In [43]:
```
A2.type_of_one() # the chirality of the complex identity
```

Out[43]: both-handed one

Let's activate now an exceptional K-algebra, the not well known "euclidean numbers" (by index "E2").
The related rectangular matrix is build up by two stacked identity matrices which are at the same time an alias for the euclidean metric operator.

In [44]:
```
A2_E = A2.A_from_multi_tab(Dict_M2A.get("E2")); A2_E,A2.ini  # 2-dim euclidean numbers
```

Out[44]:
$$\left(\begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} a_{111} & a_{112} \\ a_{121} & a_{122} \\ a_{211} & a_{212} \\ a_{221} & a_{222} \end{pmatrix}\right)$$

In [45]:
```
A2.multi_tab()     # the multiplication table of the 2-dim euclidean numbers
```

Out[45]:
$$\begin{pmatrix} g_1 & g_2 \\ g_1 & g_2 \end{pmatrix}$$

In [46]:
```
o = A2.one(); o   # the identity of the 2-dim  euclidean numbers
```

Out[46]:
$$\begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \end{pmatrix}$$

In this case the algebraic one is not the usual unit vector.

In addition this is only a left-handed one.

In [47]:
```
A2.type_of_one()
```

Out[47]:
```
left-handed one
```

Checking this out by the binary product.

In [48]:
```
q = A2.number('q');
o *A* q == q
```

Out[48]: True

The negative result by using the same identity from the right side.

In [49]:
```
q *A* o == q
```

Out[49]: False

2.2.3 Cartesian product algebras

Tessarines (https://en.wikipedia.org/wiki/Bicomplex_number) are complex numbers where the field is defined over the split-complex numbers.
Symbolically those are created as cartesian product K-algebra $\mathbb{SC} \times \mathbb{C}$.
They are building a four dimensional K-algebra over the real field $\mathbb{R}$.
Which is now the related rectangular matrix ***A*** over the reals by $\mathbb{R} \times \mathbb{A}_4$, representing the cartesian product K-algebra $\mathbb{SC} \times \mathbb{C}$?

> **Method:** .XY_cartesian(Y)
> returns the cartesian product matrix of X x Y (X = .A active K-algebra).
> The method uses:
> * X by default the active rectangular matrix for the loaded K-algebra (in the .A method)
> * Y must be entered
> > output is then X x Y

In [50]:
```
A2 = K_Algebra(2);
A2.set_algebra(A_SC);                        # set first the split-complex number as active
```

Enter now the rectangular matrix A_C for the complex numbers as input. The output is the related rectangular matrix for the tessarines A_TS.

In [51]:
```
A_TS = A2.XY_cartesian(A_C);
```

The cartesian product of two K-algebra's is not the tensor product between the two K-algebras: $\mathbb{SC} \times \mathbb{C} \neq \mathbb{SC} \otimes \mathbb{C}$

In [52]:
```
A_SC.tensor_product(A_C)==A_TS
```

Out[52]: False

In [53]:
```
A4 = K_Algebra(4);
A4.set_algebra(A_TS);                       # activate now the tessarines
```

In [54]:
```
M_TS=A4.multi_tab();M_TS                    # display the multiplication table
```

Out[54]:
$$\begin{pmatrix} g_1 & g_2 & g_3 & g_4 \\ g_2 & -g_1 & g_4 & -g_3 \\ g_3 & g_4 & g_1 & g_2 \\ g_4 & -g_3 & g_2 & -g_1 \end{pmatrix}$$

In [55]:
```
A4.is_commutative(),A4.is_associative()
```

Out[55]:
```
(commutative, associative)
```

In [56]:
```
A4.one(), A4.type_of_one()
```

Out[56]:
$$\left(\begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}, \text{both-handed one} \right)$$

Method: .non_zero()
returns the non-zero values of the weights of a K-algebra as list for big matrices.

In [57]:
```
A4.non_zero()   # the non-zero weights for the Tessarines
```

Out[57]: $[[a_{111}, 1], [a_{122}, 1], [a_{133}, 1], [a_{144}, 1], [a_{212}, 1], [a_{221}, -1], [a_{234}, 1], [a_{243}, -1], [a_{313}, 1], [a_{324}, 1], [a_{331}, 1], [a_{342}, 1], [a_{414}, 1], [a_{423}, -1], [a_{432}, 1], [a_{441}, -1]]$

In [58]:
```
p = A4.number('p'); q = A4.number('q');
p *A* q
```

Out[58]:
$$\begin{pmatrix} p_1q_1 - p_2q_2 + p_3q_3 - p_4q_4 \\ p_2q_1 + p_1q_2 + p_4q_3 + p_3q_4 \\ p_3q_1 - p_4q_2 + p_1q_3 - p_2q_4 \\ p_4q_1 + p_3q_2 + p_2q_3 + p_1q_4 \end{pmatrix}$$

Some more cartesian K-algebras to play around (be aware those can take quite a run time, therefore only listed here!))

```
* Double complexified numbers (complex numbers where the field is complex) ~ ℂ×ℂ
  A2 = K_algebra(2);
  A_C = A_from_multi_tab(Dict_M2A.get("C"));
  A_CxC = A2.XY_cartesian(A_C);
  A4 = K_algebra(4);
  set_algebra(A_CxC);

* Biquaternions (quaternions where the field is complex) ~ ℂ×ℍ
  A2 = K_algebra(2);
  A_C = A_from_multi_tab(Dict_M2A.get("C"));
  A4 = K_algebra(4);
  A_H = A_from_multi_tab(Dict_M2A.get("H"));
  A_BQ = A2.XY_cartesian(A_H);
  A8 = K_algebra(8);
  set_algebra(A_BQ);

* The E8 x E8 ~  𝕆×𝕆  64-dim K-algebra related to string theory.
  A8 = K_Algebra(8);
  A_O = A_from_multi_tab(Dict_M2A.get("O"));
  A_OxO = A8.XY_cartesian(A_O);
  A64 = K_Algebra(64);
  A64.set_algebra(A_OxO);

* The Geoffrey Dickson cartesian product K-algebra which is ℂ×ℍ×𝕆,
  a 64 dim K-algebra over the reals which relates to the U(1)xSU(2)xSU(3) group in elementary particle physics.
  A8 = K_Algebra(8);
  A_O = A_from_multi_tab(Dict_M2A.get("O"));
  set_algebra(A_BQ);  # refer to Bi-quaternions setup
  A_CQO = XY_cartesian(A_O);
  A64 = K_Algebra(64);
  A64.set_algebra(A_CQO);
```

2.2.4 The dual operators for the generator base and for the field base

For this demo the Hamilton quaternions are loaded, available as identifier "H" in the multiplication table dictionary.

In [59]:
```
A4 = K_Algebra(4);
A_H = A4.A_from_multi_tab(Dict_M2A.get("H"));
A4.non_zero()
```

Out[59]: $[[a_{111}, 1], [a_{122}, 1], [a_{133}, 1], [a_{144}, 1], [a_{212}, 1], [a_{221}, -1], [a_{234}, 1], [a_{243}, -1], [a_{313}, 1], [a_{324}, -1], [a_{331}, -1], [a_{342}, 1], [a_{414}, 1], [a_{423}, 1], [a_{432}, -1], [a_{441}, -1]]$

In [60]:
```
A4.multi_tab() # the related multiplication table of the quaternions
```

Out[60]:
$$\begin{pmatrix} g_1 & g_2 & g_3 & g_4 \\ g_2 & -g_1 & g_4 & -g_3 \\ g_3 & -g_4 & -g_1 & g_2 \\ g_4 & g_3 & -g_2 & -g_1 \end{pmatrix}$$

In [61]:
```
(A4.is_commutative(),A4.is_associative())  # check commutativity and associativity
```

Out[61]: (not commutative, associative)

In [62]:
```
A4.one(), A4.type_of_one()                # the identity & chirality of the quaternions
```

Out[62]:
$$\left(\begin{pmatrix}1\\0\\0\\0\end{pmatrix}, \text{both-handed one}\right)$$

In [63]:
```
p = A4.number('p'); q = A4.number('q');
p *A* q   #the quaternion product
```

Out[63]:
$$\begin{pmatrix}p_1q_1 - p_2q_2 - p_3q_3 - p_4q_4\\ p_2q_1 + p_1q_2 - p_4q_3 + p_3q_4\\ p_3q_1 + p_4q_2 + p_1q_3 - p_2q_4\\ p_4q_1 - p_3q_2 + p_2q_3 + p_1q_4\end{pmatrix}$$

Compared to the Sage implementation of quaternions. Here (i,j,k) are mapped to (i_1,i_2,i_3) to avoid conflicts with the existing i definition of Sagemath.

In [64]:
```
HQ.<i_1,i_2,i_3>=QuaternionAlgebra(SR,-1,-1)
```

In [65]:
```
res=HQ(p_1+p_2*i_1+p_3*i_2+p_4*i_3)*HQ(q_1+q_2*i_1+q_3*i_2+q_4*i_3)
```

In [66]:
```
matrix([expand(res.coefficient_tuple()[j]) for j in [0..3]]).T
```

Out[66]:
$$\begin{pmatrix}p_1q_1 - p_2q_2 - p_3q_3 - p_4q_4\\ p_2q_1 + p_1q_2 - p_4q_3 + p_3q_4\\ p_3q_1 + p_4q_2 + p_1q_3 - p_2q_4\\ p_4q_1 - p_3q_2 + p_2q_3 + p_1q_4\end{pmatrix}$$

Method: .H(*q)
returns the minimal hermitean operator of the K-algebra's generator base.
The method will return:
* a n x n matrix, representing the minimal H operator, if exists
* {} if not exists
* optional will return the hermitean number Hq for a given number q

Be aware of:
The minimal H operator can be identified with the regular conjugation operator only for certain K-algebras!
It is defined as the minimal dual operator for the generator base, existant under constraints!
The regular conjugation operators are usually defined in correspondence to the complex numbers.
Refer to the appendix for more details about the differences.

In [67]:
```
A4.H()                    # the minimal hermitean operator for the quaternions is here identical to the conjugation
```

Out[67]:
$$\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & -1 & 0 & 0\\ 0 & 0 & -1 & 0\\ 0 & 0 & 0 & -1\end{pmatrix}$$

In [68]:
```
A4.H(q)                   # the conjugated quaternion
```

Out[68]:
$$\begin{pmatrix}q_1\\ -q_2\\ -q_3\\ -q_4\end{pmatrix}$$

In [69]:
```
A4.H(q) *A* q             # the product of conj(q)*q = norm(q)
```

Out[69]:
$$\begin{pmatrix}q_1^2 + q_2^2 + q_3^2 + q_4^2\\ 0\\ 0\\ 0\end{pmatrix}$$

Euler's "Vier-Quadrate-Satz" (letter to Goldbach May 4, 1748): |pq| == |p| |q|

In [70]:
```
simplify(expand(A4.H((p *A* q)) *A* (p *A* q) - (A4.H(p) *A* p) *A*  (A4.H(q) *A* q)))
```

Out[70]:
$$\begin{pmatrix}0\\0\\0\\0\end{pmatrix}$$

Method: .G(*q)
returns the minimal metric operator (dual field operator) of a K-algebra (via projection onto the identity from the minimal H operator).
The minimal G operator might differ from the regular G operator.
The method will return:
* a n x n matrix, representing the minimal G operator, if exists
* {} if no minimal induced metric exists
* optional will return the covariant number Gq for a given contravariant number q

In [71]:
```
A4.G()                          # the quaternions are euclidean
```

Out[71]:
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

The covariant coordinates are here identical to the contravariant ones $q^j = q_j$:

In [72]:
```
A4.G(q)
```

Out[72]:
$$\begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{pmatrix}$$

Method: .H_R(*q)
returns the regular hermitean operator "conjugation" of the K-algebra's generator base.
The method will return:
* a n x n matrix, representing the regular H operator, if exists
* {} if not exists
* optional will return the conjugated number Hq for a given number q

In [73]:
```
A4.H_R()     # for the quaternions the regular hermitean operator is identical to the minimal one
```

Out[73]:
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$$

Method: .G_R(*q)
returns the regular metric operator (dual field operator) of a K-algebra (via projection onto the identity from the regular H operator).
The regular G operator might differ from the minimal G operator.
The method will return:
* a n x n matrix, representing the regular G operator, if exists
* {} if no regular induced metric exists
* optional will return the covariant number Gq for a given contravariant number q

In [74]:
```
A4.G_R()    # for the quaternions the regular metric is identical to the minimal metric
```

Out[74]:
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

2.2.5 The minimal Dirac Operator for a K-algebra and its algebraic constraints for existence

Method: .P_Dirac(*A)
returns the minimal Dirac operator. The method will return: a n x n symbolic matrix differential operator, representing the minimal Dirac operator for the active K-algebra, optional the minimal Dirac operator for a given A

Because Python 2 is allowing only standard characters in variable definitions (for good portability reasons), the partial derivation symbol ∂_j is here replaced by d_j. The output is only symbolically to be interpreted and needs to be transformed to the concrete diff() operator in SageMath for concrete calculations!

Method: .Cauchy_Riemann_Eq(*A)
returns the generalized Cauchy-Riemann equation in symbolic form.
The method will return:
* a partial differential system as algebraic constraints, which is required that related P_Dirac op exists
* optionally the Cauchy-Riemann system for a different A

For the demo, let's start with the complex numbers again to see the correspondencies.

In [75]:
```
A2 = K_Algebra(2);
A2.set_algebra(A_C);
```

The Dirac operator of the complex numbers is related to the **Wirtinger calculus (https://en.wikipedia.org/wiki/Wirtinger_derivatives)**.
For the complex numbers this is the Wirtinger derivation put in form as matrix operator.

In [76]:
```
A2.P_Dirac()
```

Out[76]:
$$\begin{pmatrix} \frac{1}{2}\,d_1 & \frac{1}{2}\,d_2 \\ -\frac{1}{2}\,d_2 & \frac{1}{2}\,d_1 \end{pmatrix}$$

Why is this operator here referred as Dirac operator?
In the appendix will be shown that the K-algebra of the γ-matrices (key "D4" in the dictionary) of the quantum mechanical Dirac equation for the electron results to the Dirac operator $i\gamma^j\partial_j$ by this method. A related minimal procedure is used as a template to generalize complex analysis to algebraic analysis.
The overdetermined algebraic constraints, required for the existance of the Dirac (Wirtinger) operator,
which is mediating the complex derivation of a complex function f, are the Cauchy-Riemann equations.
The listed vector must be set to a corresponding zero vector on the right side.

In [77]:
```
A2.Cauchy_Riemann_Eq()
```

Out[77]:
$$\begin{pmatrix} d_1f_1 - d_2f_2 \\ d_2f_1 + d_1f_2 \\ d_2f_1 + d_1f_2 \\ -d_1f_1 + d_2f_2 \end{pmatrix}$$

Dirac (Wirtinger) operator acting on a arbitrary complex function f gives:

In [78]:
```
f = A2.number('f'); Df = A2.P_Dirac(A_C)*f; Df
```

Out[78]:
$$\begin{pmatrix} \frac{1}{2}\,d_1f_1 + \frac{1}{2}\,d_2f_2 \\ -\frac{1}{2}\,d_2f_1 + \frac{1}{2}\,d_1f_2 \end{pmatrix}$$

Joining this with the Cauchy-Riemann constraints ($d_2f_2 \to d_1f_1$, $d_2f_1 \to -d_1f_2$) and using the (1,i) generator base returns the classic complex derivation:

In [79]:
```
r = matrix([Df.list()[0].substitute(1/2*d_2*f_2 == 1/2*d_1*f_1 ), Df.list()[1].substitute(-1/2*d_2*f_1 == 1/2*d_1*f_2 )]).T;r
```

Out[79]:
$$\begin{pmatrix} d_1f_1 \\ d_1f_2 \end{pmatrix}$$

In [80]:
```
matrix([1,I])*r
```

Out[80]:
$$\begin{pmatrix} d_1f_1 + i\,d_1f_2 \end{pmatrix}$$

Did you ever read of the Cauchy-Riemann equations of the vector product?
This concept, borrowed from quantum mechanics, will do the job and does not need even an algebraic identity!
Now here they are:

In [81]:
```
A3 = K_Algebra(3);
A3.set_algebra(A_VP3);
```

Because the vector product is not commutative there is now a left- and right-handed Dirac operator.
$P_{Dirac,left} = -P_{Dirac,right}$

In [82]:
```
A3.P_Dirac(),A3.P_Dirac(A3.B())
```

Out[82]:
$$\left(\begin{pmatrix} 0 & -\frac{1}{2}\,d_3 & \frac{1}{2}\,d_2 \\ \frac{1}{2}\,d_3 & 0 & -\frac{1}{2}\,d_1 \\ -\frac{1}{2}\,d_2 & \frac{1}{2}\,d_1 & 0 \end{pmatrix}, \begin{pmatrix} 0 & \frac{1}{2}\,d_3 & -\frac{1}{2}\,d_2 \\ -\frac{1}{2}\,d_3 & 0 & \frac{1}{2}\,d_1 \\ \frac{1}{2}\,d_2 & -\frac{1}{2}\,d_1 & 0 \end{pmatrix}\right)$$

Both existence constraints for related chiral Dirac operators are here identical!

In [83]:
```
A3.Cauchy_Riemann_Eq(A_VP3), A3.Cauchy_Riemann_Eq(A3.B(A_VP3))
```

Out[83]:
$$\left(\begin{pmatrix} 2\,d_1f_1 \\ d_2f_1 + d_1f_2 \\ d_3f_1 + d_1f_3 \\ d_2f_1 + d_1f_2 \\ 2\,d_2f_2 \\ d_3f_2 + d_2f_3 \\ d_3f_1 + d_1f_3 \\ d_3f_2 + d_2f_3 \\ 2\,d_3f_3 \end{pmatrix}, \begin{pmatrix} 2\,d_1f_1 \\ d_2f_1 + d_1f_2 \\ d_3f_1 + d_1f_3 \\ d_2f_1 + d_1f_2 \\ 2\,d_2f_2 \\ d_3f_2 + d_2f_3 \\ d_3f_1 + d_1f_3 \\ d_3f_2 + d_2f_3 \\ 2\,d_3f_3 \end{pmatrix}\right)$$

This new concept is now applicable to any K-algebra.
Let's move on to the Tessarines.

In [84]:
```
A4 = K_Algebra(4);
A4.set_algebra(A_TS);
```

In [85]:
```
A4.P_Dirac(),A4.Cauchy_Riemann_Eq()                    # the Dirac op & Cauchy-Riemann system for the Tessarines
```

Out[85]:
$$\left(\begin{pmatrix} \frac{1}{4}d_1 & \frac{1}{4}d_2 & \frac{1}{4}d_3 & \frac{1}{4}d_4 \\ -\frac{1}{4}d_2 & \frac{1}{4}d_1 & -\frac{1}{4}d_4 & \frac{1}{4}d_3 \\ \frac{1}{4}d_3 & \frac{1}{4}d_4 & \frac{1}{4}d_1 & \frac{1}{4}d_2 \\ -\frac{1}{4}d_4 & \frac{1}{4}d_3 & -\frac{1}{4}d_2 & \frac{1}{4}d_1 \end{pmatrix}, \begin{pmatrix} 3d_1f_1 - d_2f_2 - d_3f_3 - d_4f_4 \\ d_2f_1 + 3d_1f_2 + d_4f_3 - d_3f_4 \\ -d_3f_1 - d_4f_2 + 3d_1f_3 - d_2f_4 \\ d_4f_1 - d_3f_2 + d_2f_3 + 3d_1f_4 \\ 3d_2f_1 + d_1f_2 - d_4f_3 + d_3f_4 \\ -d_1f_1 + 3d_2f_2 - d_3f_3 - d_4f_4 \\ -d_4f_1 + d_3f_2 + 3d_2f_3 + d_1f_4 \\ -d_3f_1 - d_4f_2 - d_1f_3 + 3d_2f_4 \\ 3d_3f_1 - d_4f_2 - d_1f_3 - d_2f_4 \\ d_4f_1 + 3d_3f_2 + d_2f_3 - d_1f_4 \\ -d_1f_1 - d_2f_2 + 3d_3f_3 - d_4f_4 \\ d_2f_1 - d_1f_2 + d_4f_3 + 3d_3f_4 \\ 3d_4f_1 + d_3f_2 - d_2f_3 + d_1f_4 \\ -d_3f_1 + 3d_4f_2 - d_1f_3 - d_2f_4 \\ -d_2f_1 + d_1f_2 + 3d_4f_3 + d_3f_4 \\ -d_1f_1 - d_2f_2 - d_3f_3 + 3d_4f_4 \end{pmatrix}\right)$$

Also not usual K-algebra's like the logical "AND" do have their related Cauchy-Riemann constraints now.

In [86]:
```
A2 = K_Algebra(2);
A_AND = A2.A_from_multi_tab(Dict_M2A.get("AND")); A_AND,A2.ini
```

Out[86]:
$$\left(\begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}, \begin{pmatrix} a_{111} & a_{112} \\ a_{121} & a_{122} \\ a_{211} & a_{212} \\ a_{221} & a_{222} \end{pmatrix}\right)$$

In [87]:
```
A2.multi_tab()  # the logical multiplication table of the "AND" where g_1 ~ 0 and g_2 ~ 1
```

Out[87]:
$$\begin{pmatrix} g_1 & g_1 \\ g_1 & g_2 \end{pmatrix}$$

In [88]:
```
A2.P_Dirac(), A2.Cauchy_Riemann_Eq() # the Dirac operator of the "AND" & Cauchy-Riemann system
```

Out[88]:
$$\left(\begin{pmatrix} \frac{1}{3}d_1 + \frac{2}{3}d_2 & -\frac{1}{3}d_2 \\ \frac{1}{3}d_1 - \frac{1}{3}d_2 & \frac{2}{3}d_2 \end{pmatrix}, \begin{pmatrix} \frac{2}{3}d_1f_1 - \frac{2}{3}d_2f_1 - \frac{2}{3}d_2f_2 \\ 2d_1f_2 \\ -\frac{2}{3}d_1f_1 + \frac{2}{3}d_2f_1 + \frac{2}{3}d_2f_2 \\ -\frac{2}{3}d_1f_1 + \frac{2}{3}d_2f_1 + \frac{2}{3}d_2f_2 \end{pmatrix}\right)$$

In [89]:
```
A2.is_commutative(),A2.is_associative()
```

Out[89]:
(commutative, associative)

In [90]:
```
A2.one(),A2.type_of_one()   # the logic "AND" has a both-handed identity
```

Out[90]:
$$\left(\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \text{both-handed one}\right)$$

In [91]:
```
A2.H(),A2.G()   # but no minimal conjugation and no minimal metric
```

Out[91]:
({}, {})

In [92]:
```
A2.H_R(),A2.G_R() # and also no regular conjugation and no regular metric
```

Out[92]:
({}, {})

In [93]:
```
p = A2.number('p'); q = A2.number('q');  # multiplication of the "AND"
p *A* q
```

Out[93]:
$$\begin{pmatrix} p_1q_1 + p_2q_1 + p_1q_2 \\ p_2q_2 \end{pmatrix}$$

3. Superscript operators B, I, R, T

The generator free formalism is requiring the

- **R**ectangular matrix calculus in combination with the
- **K**roneckerproduct including following additional
- **S**uper script operators
  (shortly reffered as **RKS** notation).

Following additional four super script symbols are used to act on the rectangular matrix ***A*** for a K-algebra. These are:

- $\boldsymbol{A}^B$: where B is used as alias of the index shift operator of the bilateral exchanged first two indices of the triple index
- $\boldsymbol{A}^I$: where I is used as alias of the standard pseudoinverse() method for matrices
- $\boldsymbol{A}^R$: where R is used as alias of the index shift operator of the right shifted indices of the triple index
- $\boldsymbol{A}^T$: where T is used as alias of the standard transpose() method for matrices

Note,

- that B, R acting on quadratic matrices are reduced to the transpose() operator, according to their logic,
- that I acting on non-singular quadratic matrices are giving the ordinary inverse of a matrix,
- that I acting on singular matrices is also possible by using the minimal solution,
- that I acting on a zero vector or zero matrix returns the transposed zero vector and zero matrix!

The definition of user defined superscript operators is currently not possible in SageMath 8.8 (according to the author's knowledge).
Therefore these superscript operators are realized in Sage as class methods.
From the programmer's view that is anyway a better way to avoid conflicts with existing Sage definitions.
For example the ()^B operator is created as class method .B()!

However, the RKS notation is more convenient for a concrete documentation.
The I and T operators are already commonly in use as superscripts, only the B, R shift operators are joining.
The sequence of these operators needs to be taken into account and the correct bracketing!
For example A^{BRI} means ((A^B)^R)^I and in general these operators do not commute BR != RB.
Beside T is commuting with all of them and only I commutes with B.

For the demonstration of the actions let's define an arbitrary 3-dim instance of a K-algebra:

In [94]:
```
A3 = K_Algebra(3);
```

Method: .B(*A)
returns the bilateral shifted rectangular matrix A.
The method will return:
* by default the bilateral shifted sibling of the active matrix a(j,k,l) -> a(k,j,l)
* optional for a given matrix A

In the RKS notation: $(a_{jkl})^B = a_{kjl}$

Here for comparison the original rectangular matrix ***A*** with the bilateral shifted matrix $\boldsymbol{A}^B$:

In [95]:
```
A3.A, A3.B()
```

Out[95]:
$$\left(\begin{pmatrix} a_{111} & a_{112} & a_{113} \\ a_{121} & a_{122} & a_{123} \\ a_{131} & a_{132} & a_{133} \\ a_{211} & a_{212} & a_{213} \\ a_{221} & a_{222} & a_{223} \\ a_{231} & a_{232} & a_{233} \\ a_{311} & a_{312} & a_{313} \\ a_{321} & a_{322} & a_{323} \\ a_{331} & a_{332} & a_{333} \end{pmatrix}, \begin{pmatrix} a_{111} & a_{112} & a_{113} \\ a_{211} & a_{212} & a_{213} \\ a_{311} & a_{312} & a_{313} \\ a_{121} & a_{122} & a_{123} \\ a_{221} & a_{222} & a_{223} \\ a_{321} & a_{322} & a_{323} \\ a_{131} & a_{132} & a_{133} \\ a_{231} & a_{232} & a_{233} \\ a_{331} & a_{332} & a_{333} \end{pmatrix}\right)$$

Method: .I(*A)
returns the pseudoinverse().
The method will return:
* by default the pseudoinverse of the active matrix .A
* optional for a given matrix A

For a given rectangular matrix, here the complex numbers by defined related rectangular matrix A_C:

In [96]:
```
A2.I(A_C)       # the pseudoinverse of A_C
```

Out[96]:
$$\begin{pmatrix} \frac{1}{2} & 0 & 0 & -\frac{1}{2} \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}$$

Method: .R(*A)
returns the rectangular matrix A which is right shifted in the triple indices.
The method will return:
* by default the right shifted sibling of the active matrix .A; a(j,k,l) -> a(l,j,k)
* optional for a given matrix A

The right shifted (rotate thru carry) operator R is acting on the triple indices as follows: $(a_{jkl})^R = a_{ljk}$

Here for comparison the original rectangular matrix $\boldsymbol{A}$ with the right shifted matrix $\boldsymbol{A}^R$:

In [97]:
```
A3.A, A3.R()
```

Out[97]:
$$\left(\begin{pmatrix} a_{111} & a_{112} & a_{113} \\ a_{121} & a_{122} & a_{123} \\ a_{131} & a_{132} & a_{133} \\ a_{211} & a_{212} & a_{213} \\ a_{221} & a_{222} & a_{223} \\ a_{231} & a_{232} & a_{233} \\ a_{311} & a_{312} & a_{313} \\ a_{321} & a_{322} & a_{323} \\ a_{331} & a_{332} & a_{333} \end{pmatrix}, \begin{pmatrix} a_{111} & a_{211} & a_{311} \\ a_{112} & a_{212} & a_{312} \\ a_{113} & a_{213} & a_{313} \\ a_{121} & a_{221} & a_{321} \\ a_{122} & a_{222} & a_{322} \\ a_{123} & a_{223} & a_{323} \\ a_{131} & a_{231} & a_{331} \\ a_{132} & a_{232} & a_{332} \\ a_{133} & a_{233} & a_{333} \end{pmatrix}\right)$$

Method: .T(*A)
a shortcut to the .transpose() function.
The method will return:
* by default the transposed of the active matrix .A
* optional for a given matrix A

In [98]:
```
A3.T()
```

Out[98]:
$$\begin{pmatrix} a_{111} & a_{121} & a_{131} & a_{211} & a_{221} & a_{231} & a_{311} & a_{321} & a_{331} \\ a_{112} & a_{122} & a_{132} & a_{212} & a_{222} & a_{232} & a_{312} & a_{322} & a_{332} \\ a_{113} & a_{123} & a_{133} & a_{213} & a_{223} & a_{233} & a_{313} & a_{323} & a_{333} \end{pmatrix}$$

The two shift operators B, R are already forming a basis for all six available permutations including the identity.

All possible six permutations are then $\boldsymbol{A}$, $\boldsymbol{A}^B$, $\boldsymbol{A}^R$, $\boldsymbol{A}^{RB}$,$\boldsymbol{A}^{BR}$, $\boldsymbol{A}^{RR}$.

In [99]:
```
A3.A, A3.B(), A3.R(), A3.B(A3.R()), A3.R(A3.B()),A3.R(A3.R())
```

Out[99]:
$$\left(\begin{pmatrix} a_{111} & a_{112} & a_{113} \\ a_{121} & a_{122} & a_{123} \\ a_{131} & a_{132} & a_{133} \\ a_{211} & a_{212} & a_{213} \\ a_{221} & a_{222} & a_{223} \\ a_{231} & a_{232} & a_{233} \\ a_{311} & a_{312} & a_{313} \\ a_{321} & a_{322} & a_{323} \\ a_{331} & a_{332} & a_{333} \end{pmatrix}, \begin{pmatrix} a_{111} & a_{112} & a_{113} \\ a_{211} & a_{212} & a_{213} \\ a_{311} & a_{312} & a_{313} \\ a_{121} & a_{122} & a_{123} \\ a_{221} & a_{222} & a_{223} \\ a_{321} & a_{322} & a_{323} \\ a_{131} & a_{132} & a_{133} \\ a_{231} & a_{232} & a_{233} \\ a_{331} & a_{332} & a_{333} \end{pmatrix}, \begin{pmatrix} a_{111} & a_{211} & a_{311} \\ a_{112} & a_{212} & a_{312} \\ a_{113} & a_{213} & a_{313} \\ a_{121} & a_{221} & a_{321} \\ a_{122} & a_{222} & a_{322} \\ a_{123} & a_{223} & a_{323} \\ a_{131} & a_{231} & a_{331} \\ a_{132} & a_{232} & a_{332} \\ a_{133} & a_{233} & a_{333} \end{pmatrix}, \begin{pmatrix} a_{111} & a_{211} & a_{311} \\ a_{121} & a_{221} & a_{321} \\ a_{131} & a_{231} & a_{331} \\ a_{112} & a_{212} & a_{312} \\ a_{122} & a_{222} & a_{322} \\ a_{132} & a_{232} & a_{332} \\ a_{113} & a_{213} & a_{313} \\ a_{123} & a_{223} & a_{323} \\ a_{133} & a_{233} & a_{333} \end{pmatrix}, \begin{pmatrix} a_{111} & a_{121} & a_{131} \\ a_{112} & a_{122} & a_{132} \\ a_{113} & a_{123} & a_{133} \\ a_{211} & a_{221} & a_{231} \\ a_{212} & a_{222} & a_{232} \\ a_{213} & a_{223} & a_{233} \\ a_{311} & a_{321} & a_{331} \\ a_{312} & a_{322} & a_{332} \\ a_{313} & a_{323} & a_{333} \end{pmatrix}, \begin{pmatrix} a_{111} & a_{121} & a_{131} \\ a_{211} & a_{221} & a_{231} \\ a_{311} & a_{321} & a_{331} \\ a_{112} & a_{122} & a_{132} \\ a_{212} & a_{222} & a_{232} \\ a_{312} & a_{322} & a_{332} \\ a_{113} & a_{123} & a_{133} \\ a_{213} & a_{223} & a_{233} \\ a_{313} & a_{323} & a_{333} \end{pmatrix}\right)$$

Having the two new shift operators B, R available, certain (not well known) Kronecker identities can be derived which are heavily used in the following derivation paths:

$\boldsymbol{A}^T(p \otimes q) = \boldsymbol{A}^{BT}(q \otimes p)$

$\text{vec}(\boldsymbol{A}^T(p \otimes E)) = \boldsymbol{A}^R p$

Let's shortly exemplary check that identities with the here given arbitrary 3-dim K-algebras.

In [100]:
```
p = A3.number('p'); q = A3.number('q'); A = A3.ini;
A.T*(p.tensor_product(q)) == (A3.B(A).T)*(q.tensor_product(p))
# exemplary in the 3-dim case but trivially valid for n-dim because B shifts the first two indices
```

Out[100]: True

In [101]:
```
E = A3.E();
A3.vec(A.T*(p.tensor_product(E)))== A3.R(A)*p
```

Out[101]: True

Appendix

A.1 Determine the identity (the algebraic one) of a K-algebra

Let $\tilde{o}$ the be searched for left action one (identity) of a K-Algebra $\mathbb{A}$ of dim =n, with the property:

$\tilde{o} \otimes_{\mathbb{A}} \tilde{q} = \tilde{q} \quad \forall\, \tilde{q} \neq 0$

$\Leftrightarrow \boldsymbol{A}^T(o \otimes q) = q = \boldsymbol{A}^T(o \otimes E)\, q$ with E being the identitymatrix(n)

Because the above matrix equation in q needs to be fullfilled for any q the following relation must be valid:

$\boldsymbol{A}^T(o \otimes E) = E$

Using the vec() operator on each side and in the chapter before mentioned Kronecker identity $\text{vec}(\boldsymbol{A}^T(p \otimes E)) = \boldsymbol{A}^R p$ is yielding:

$\text{vec}(\boldsymbol{A}^T(o \otimes E)) = \text{vec}(E) = \boldsymbol{A}^R o$

This is a classical overdetermined matrix-vector problem.
The related minimal solution according to an arbitrary pseudo norm of type $q^T G q$ with $\det(G) = \pm 1$ is only possible,
if certain algebraic constraints are additionally fulfilled.

$o_{left} = \boldsymbol{A}^{RI}\text{vec}(E) \Leftrightarrow \boldsymbol{A}^R\boldsymbol{A}^{RI}\text{vec}(E) == \text{vec}(E)$

The right one is now quickly done using the shift operator equality:

$\boldsymbol{A}^T(p \otimes q) = \boldsymbol{A}^{BT}(q \otimes p)$

$o_{right} = \boldsymbol{A}^{BRI}\text{vec}(E) \Leftrightarrow \boldsymbol{A}^{BR}\boldsymbol{A}^{BRI}\text{vec}(E) == \text{vec}(E)$

If the left-one and right-one are both existing then necessarily these are identical. In this case the identity is called "both-handed" one.

Summarizing, there can be only four cases for chirality:

- no existant one (because both algebraic constraints are not fulfilled)
- both-handed one (both algebraic constraints are fulfilled)
- left-handed one (only the left-handed constraints are fulfilled)
- right-handed one (only the right-handed constraints are fulfilled)

A.2 Cartesian product of K-algebras

Given to generator bases g_x, h_y of two related K-algebras $\mathbb{X}, \mathbb{Y}$ with related dimensions (x, y).
The cartesian product $\mathbb{X} \times \mathbb{Y} = \mathbb{Z}$ is then defined by:
$(g_x \otimes h_y) \otimes (g_x \otimes h_y) = \boldsymbol{Z}\,(g_x \otimes h_y)$

With a generalization of the B shift operator for two different dimensional bases with the property:

$B\,(g_x \otimes h_y) = (h_y \otimes g_x)$ with $B\,(x, y) = \sum_{j=1}^{y} e_j \otimes E_x \otimes e_j^T$

implies, (where E_x, E_y are the x, y dimensional identity matrices):
$(g_x \otimes g_y) \otimes (g_x \otimes g_y) = (E_x \otimes B \otimes E_y)\,(\boldsymbol{X} \otimes \boldsymbol{Y})\,(g_x \otimes g_y)$

Resulting for the searched rectangular matrix $\boldsymbol{Z}$ to:

$\boldsymbol{Z} = (E_x \otimes B \otimes E_y)\,(\boldsymbol{X} \otimes \boldsymbol{Y})$

Method: .B_cartesian(m)
The method will return:
* the commutator operator for a product base of two different generator bases B(gx⊗gy) = (gy⊗gx)
where gx represents the generators of the active K-algebra and gy the multiplicant generators of dimension m

In [102]:
```
A2.B_cartesian(4)   # the B operator for a 2-dim active K-algebra over a 4-dim K-algebra
```

Out[102]:
$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

The cartesian product is generally not commutative. Creating the cartesian product $\mathbb{X} \times \mathbb{Y}$ or vice versa $\mathbb{Y} \times \mathbb{X}$ is leading to different K-algebras in most cases. Exemplary, $\mathbb{C} \times \mathbb{H}$ are the biquaternions, these are quaternions where the field are complex numbers. $\mathbb{H} \times \mathbb{C}$ is describing complex numbers where the field are quaternions instead of the reals. The here given method does not require that a K-algebra in the cartesian product is a field! Minimum requirement is, that both multiplicands in the cartesian product are K-algebras, which can be represented by a rectangular matrix each.

For demonstration the the biquaternions $\mathbb{C} \times \mathbb{H}$ are compared with the $\mathbb{H} \times \mathbb{C}$ K-algebra:

In [103]:
```
A2=K_Algebra(2); A4=K_Algebra(4);  # set the instances for ℂ, ℍ
```

In [104]:
```
A_C = A2.A_from_multi_tab(Dict_M2A.get("C"));
A_H = A4.A_from_multi_tab(Dict_M2A.get("H"));
A_BQ = A2.XY_cartesian(A_H); # ℂ×ℍ
A_HC = A4.XY_cartesian(A_C); # ℍ×ℂ
A_BQ == A_HC
```

Out[104]: False

A.3 Determine the minimal dual operators of a K-algebra

A dual operator is converting a given base to the related inverse base (dual). The multiplication of both bases is yielding the identity of the underlying algebra. Therefore a pre-requisite of the existance of a dual base is the algebraic identity. For the field this is the field "1" but for the generators of a K-algebra the algebraic identity is not always existant.

Starting with the metric operator G which is mediating the lifting and lowering of indices in tensor notation.
Using the symbol † as hermitean dual operator of a real field.
The restriction on a real field is already sufficient, without losing generality because any cartesian product K-algebra,
for example over a complex field, can be mapped to a higher dimensional K-algebra over the reals.

$q^j = G^{jk}\ q_j$ or in vector notation $q^\dagger = q^T G$

The dual field base of the unit vectors e^j to e_j is defined as $e^j e_k = \delta^j_k$

This is mediating the Riemannian pseudo norm:

$q^\dagger\ q = q^T\ G\, q$ = norm(q)·1

where the field 1 is normally left out but here explicitely noted to see later the correspondence with the related generator dual base definitions.

Same approach will be now used for a generator base vector g for a K-algebra.

For the dual base $g^\dagger$ follows, equivalent to the field definition before, with H playing the role of G:

$g^\dagger = g^T H$

leading to, where $\tilde{o}$ (the algebraic one), is taking over the role of the field one:

$g^\dagger\ g = g^T\ H\, g$= norm($g$) · $\tilde{o}$

As for the metric operator, the invariance of H to a linear field transformation of the generators is required.

$g' = M\ g$ with $M^T H M = H$ and M being a quadratic matrix over the field

The related K-algebras are building an equivalency class by: $\boldsymbol{A}' = (M \otimes M)\boldsymbol{A}\ M$

The involution requirement for the generators $g^{\dagger\dagger} = g$ triggers:

$HHg = g$

But because the searched for operator H needs to be invariant under linear transformations, this can be only fullfilled, if already:

$H^2 = E \Rightarrow H = H^I$ with det(H) = ±1

Finally the inverse (dual) base for the generators can be defined by:

$g^I := g^\dagger \cdot \text{norm}(g)^{-1} \Rightarrow g^I\ g = \tilde{o}$

This is the same approach as for a pseudoinverse of a vector.

It follows then from $g^{II} = g = H^I\ H^T\ g \Rightarrow H = H^T$

If there exists an algebraic one $\tilde{o}$, the minimal operator H can be derived from the equation $g^T\ H\, g$ = norm(g) · $\tilde{o}$.

$g^T\ H\, g = (g^T \otimes g^T)$ vec(H) = $g^T \boldsymbol{A}^T$ vec(H) = norm(g)·$g^T o$

This underdetermined equation for H is fullfilled "minimally", under the following constraints:

vec(H) = norm(g)·$\boldsymbol{A}^{IT} o \Leftrightarrow \boldsymbol{A}^{IT}\boldsymbol{A}^T$vec($H$) == vec($H$)

What is left is the determination of the norm(g) factor.
For this purpose the hermitean basis product is an outgoing point:

$g^\dagger \otimes g^\dagger = g^\dagger\ \boldsymbol{A}^\dagger = (g^T \otimes g^T)(H \otimes H) = g^T \boldsymbol{A}^T (H \otimes H) = g^T H\ \boldsymbol{A}^\dagger \Rightarrow \boldsymbol{A}^\dagger = H\ \boldsymbol{A}^T\ (H \otimes H)$

Same doing now for the inverse (dual) generator base leads to:

$\boldsymbol{A}^I = \boldsymbol{A}^\dagger \cdot \text{norm}(g)^{-1}$

For a real field applies $\boldsymbol{A}^\dagger = \boldsymbol{A}^T$.

Using the trace function yields the norm(g) factor with n being the dimension of the K-algebra:

norm(g) = trace($\boldsymbol{A}^T\boldsymbol{A}$)/$n$

Having available all means, the generic inverse and the generic hermitean number in the classical representation can be defined as:

$\tilde{q}^I$:= $g^I q$ and also $\tilde{q}^\dagger$:= $g^\dagger q$

What is now the difference of the minimal hermitean operator for the generator base regarding to the regular conjugation?

The standard procedure to define a regular conjugation operator is sticking to the following properties,
used as templates from the complex numbers (Re() is the real part):

$\tilde{q}^\dagger + \tilde{q}$ = 2·Re($\tilde{q}$)·$\tilde{o}$ ∧ ($\tilde{q}^\dagger \tilde{q}$ = norm(q)·$\tilde{o}$ ∨ $\tilde{q}\,\tilde{q}^\dagger$ = norm(q)·$\tilde{o}$)

But any regular conjugation operator, fulfilling these relations is not automatically minimal and vice versa! The minimal H operator is unique defined by the generator base g, by $g^\dagger\ g$ = norm(g) · $\tilde{o}$ whether the regular conjugation operator is generally not minimal. For the complex numbers, the quaternions, the octonions the minimal H operator is identical to the regular conjugation.

Using the split-complex numbers to understand the difference.

In [105]:
```
A2 = K_Algebra(2);
A2.set_algebra(A_SC);
```

Derive first the regular conjugation and metric by looking for the inverse p to q:

In [106]:
```
p = A2.number('p'); q = A2.number('q');
solve((A2.multiply(p,q)-A2.one()).list(), p.list())
```

Out[106]:
$$\left[\left[p_1 = \frac{q_1}{q_1^2 - q_2^2}, p_2 = -\frac{q_2}{q_1^2 - q_2^2}\right]\right]$$

In the case of these fractions, the numerator is at the top defines the conjugation and the common denominator at the bottom the metric. In the case of the split-complex numbers conjugation and metric operator are therefore identical.
Let H_R, G_R the relating regular operators for the split-complex numbers:

In [107]:
```
H_R = A2.H_R(); G_R = A2.G_R(); H_R, G_R
```

Out[107]:
$$\left(\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}\right)$$

Checking this out by using $q^\dagger \otimes q$ on the left side which should result to $\tilde{o}$ ($q^T G q$) on the right side, proves the correct setting for H_R, G_R:

In [108]:
```
A2.multiply((H_R*q),q), A2.one()*(q.T*G_R*q)
```

Out[108]:
$$\left(\begin{pmatrix} q_1^2 - q_2^2 \\ 0 \end{pmatrix}, \begin{pmatrix} q_1^2 - q_2^2 \\ 0 \end{pmatrix}\right)$$

Checking the minimal condition $g^\dagger g = n \cdot \tilde{o}$ for the regular H_R and for the minimial A2.H() is here giving a negative result for the regular H_R operator!
Only the minimal A2.H() operator is fullfilling that condition.

In [109]:
```
A_SC.T * A2.vec(H_R), A_SC.T * A2.vec(A2.H())
```

Out[109]:
$$\left(\begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} 2 \\ 0 \end{pmatrix}\right)$$

Consequently the real part function, defined by $\frac{1}{2}(\tilde{q}^\dagger + \tilde{q})$ is only well defined for the regular H_R operator.
The minimal H operator is returning a vector in the case of the split-complex numbers, instead of a projected value to the algebraic one.The real plane is here rotated to a "false" real plane or "false one", mediating two conserved values under transformation instead of one value as in the regular form. The "false" one is not returning itself by self multiplication, instead of the "false" one expands for the split complex numbers by self multiplication.

In [110]:
```
1/2*((H_R*q) + q), 1/2*(A2.H(q)+q)
```

Out[110]:
$$\left(\begin{pmatrix} q_1 \\ 0 \end{pmatrix}, \begin{pmatrix} q_1 \\ q_2 \end{pmatrix}\right)$$

Same differences are for the relation $\tilde{q}^\dagger \otimes \tilde{q}$. The regular H_R operator gives a pseudo euclidean metric for the split-complex numbers, whether the minimal H operator outputs the standard euclidean metric. In other words the left pseudo-euclidean metric is not the minimal metric for the split-complex numbers!

In [111]:
```
A2.multiply((H_R*q),q), A2.multiply(A2.H(q),q)
```

Out[111]:
$$\left(\begin{pmatrix} q_1^2 - q_2^2 \\ 0 \end{pmatrix}, \begin{pmatrix} q_1^2 + q_2^2 \\ 2\,q_1 q_2 \end{pmatrix}\right)$$

A.4 Determine the minimal Dirac operator and existence constraints

The original Dirac operator P_{Dirac} is contained in the Dirac equation of the electron as follows:

$P_{Dirac} \cdot \psi = m \cdot \psi$ whereas $P_{Dirac} = \sum_{j=1}^{4} i \cdot \gamma^j \cdot \partial_j$

The Dirac equation is the eigenvalue equation of this operator (eigenvalue=m to eigenstates =ψ).
This 4x4 differential operator is connected with the K-algebra of the γ-matrices.

In [112]:
```
Gamma_1 = matrix(SR,4,4,[[1, 0, 0, 0], [0, 1, 0, 0], [0, 0, -1, 0], [0, 0, 0, -1]]);
Gamma_2 = matrix(SR,4,4,[[0, 0, 0, 1], [0, 0, 1, 0], [0, -1, 0, 0], [-1, 0, 0, 0]]);
Gamma_3 = matrix(SR,4,4,[[0, 0, 0, -I], [0, 0, I, 0], [0, I, 0, 0], [-I, 0, 0, 0]]);
Gamma_4 = matrix(SR,4,4,[[0, 0, 1, 0], [0, 0, 0, -1], [-1, 0, 0, 0], [0, 1, 0, 0]]);
```

In [113]:
```
Gamma_1,Gamma_2,Gamma_3,Gamma_4
```

Out[113]:
$$\left(\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ -1 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 0 & -i \\ 0 & 0 & i & 0 \\ 0 & i & 0 & 0 \\ -i & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}\right)$$

Sum up every summand gives the Dirac operator explicitly as matrix:

In [114]:
```
var(','.join(['d'+'_%d' % (j) for j in [1..4]]));
P_Dirac = (Gamma_1*d_1+Gamma_2*d_2+Gamma_3*d_3+Gamma_4*d_4)*I; P_Dirac
```

Out[114]:
$$\begin{pmatrix} i\,d_1 & 0 & i\,d_4 & i\,d_2 + d_3 \\ 0 & i\,d_1 & i\,d_2 - d_3 & -i\,d_4 \\ -i\,d_4 & -i\,d_2 - d_3 & -i\,d_1 & 0 \\ -i\,d_2 + d_3 & i\,d_4 & 0 & -i\,d_1 \end{pmatrix}$$

Loading now the underlying K-algebra for the Dirac operator.

In [115]:
```
A4 = K_Algebra(4);
A_D4 = A4.A_from_multi_tab(Dict_M2A.get("D4"));  A_D4.T   #display in transposed form
```

Out[115]:
$$\begin{pmatrix} -\frac{1}{2}i & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2}i & 0 & \frac{1}{2}i & -\frac{1}{2} & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{2}i & 0 & 0 & 0 & 0 & \frac{1}{2}i & \frac{1}{2} & 0 & 0 & 0 & 0 & -\frac{1}{2}i \\ 0 & 0 & 0 & -\frac{1}{2}i & 0 & -\frac{1}{2}i & \frac{1}{2} & 0 & \frac{1}{2}i & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{2}i & -\frac{1}{2} & 0 & 0 & 0 & 0 & \frac{1}{2}i & 0 & 0 & 0 & 0 & \frac{1}{2}i & 0 & 0 & 0 \end{pmatrix}$$

Check that active K-algebra D4 is giving the original Dirac operator:

In [116]:
```
A4.P_Dirac() == P_Dirac
```

Out[116]: True

That the K-algebra D4 is not commutative is not a suprise, because the generators of the K-algebra, the γ-matrices, do not commute.
But D4 is not associative, because the four γ-matrices are not forming a complete base for the generators by themselves! A complete base is only possible by the 16 γ-matrices, including the identity matrix as defined in the Dirac theory. In this 32 dim case over the reals the related K-algebra will be associative!
So the D4 K-algebra which is inducing the Dirac operator, cannot be mapped itself to the quadratic matrix domain!

In [117]:
```
A4.is_commutative(),A4.is_associative()
```

Out[117]: (not commutative, not associative)

In [118]:
```
A4.one()    # no identity for D4 exists
```

Out[118]: {}

The left and the right existance constraints for P_{Dirac} are significally different.

In [119]:
```
A4.Cauchy_Riemann_Eq(), A4.Cauchy_Riemann_Eq(A4.B())
```

Out[119]:

$$\left(\begin{pmatrix} -\frac{1}{4}d_1f_1+\frac{1}{4}d_4f_3+\frac{1}{4}d_2f_4-\frac{1}{4}i\,d_3f_4 \\ -\frac{1}{4}d_1f_2+\frac{1}{4}d_2f_3+\frac{1}{4}i\,d_3f_3-\frac{1}{4}d_4f_4 \\ \frac{1}{4}d_4f_1+\frac{1}{4}d_2f_2-\frac{1}{4}i\,d_3f_2-\frac{1}{4}d_1f_3 \\ \frac{1}{4}d_2f_1+\frac{1}{4}i\,d_3f_1-\frac{1}{4}d_4f_2-\frac{1}{4}d_1f_4 \\ -\frac{1}{4}d_2f_1+\frac{1}{4}i\,d_3f_1-\frac{1}{4}d_4f_2+\frac{1}{4}d_1f_4 \\ \frac{1}{4}d_4f_1-\frac{1}{4}d_2f_2-\frac{1}{4}i\,d_3f_2+\frac{1}{4}d_1f_3 \\ \frac{1}{4}d_1f_2-\frac{1}{4}d_2f_3+\frac{1}{4}i\,d_3f_3-\frac{1}{4}d_4f_4 \\ \frac{1}{4}d_1f_1+\frac{1}{4}d_4f_3-\frac{1}{4}d_2f_4-\frac{1}{4}i\,d_3f_4 \\ \frac{1}{4}i\,d_2f_1-\frac{3}{4}d_3f_1-\frac{1}{4}i\,d_4f_2+\frac{1}{4}i\,d_1f_4 \\ -\frac{1}{4}i\,d_4f_1-\frac{1}{4}i\,d_2f_2-\frac{3}{4}d_3f_2-\frac{1}{4}i\,d_1f_3 \\ \frac{1}{4}i\,d_1f_2+\frac{1}{4}i\,d_2f_3-\frac{3}{4}d_3f_3-\frac{1}{4}i\,d_4f_4 \\ -\frac{1}{4}i\,d_1f_1-\frac{1}{4}i\,d_4f_3-\frac{1}{4}i\,d_2f_4-\frac{3}{4}d_3f_4 \\ -\frac{1}{4}d_4f_1+\frac{1}{4}d_2f_2-\frac{1}{4}i\,d_3f_2+\frac{1}{4}d_1f_3 \\ -\frac{1}{4}d_2f_1-\frac{1}{4}i\,d_3f_1-\frac{1}{4}d_4f_2-\frac{1}{4}d_1f_4 \\ \frac{1}{4}d_1f_1-\frac{1}{4}d_4f_3+\frac{1}{4}d_2f_4-\frac{1}{4}i\,d_3f_4 \\ -\frac{1}{4}d_1f_2-\frac{1}{4}d_2f_3-\frac{1}{4}i\,d_3f_3-\frac{1}{4}d_4f_4 \end{pmatrix}, \begin{pmatrix} -\frac{3}{8}d_1f_1+\frac{1}{8}d_2f_2-\frac{1}{8}d_3f_3-\frac{1}{8}d_4f_4 \\ -\frac{1}{2}d_1f_2 \\ -\frac{1}{8}d_3f_1+\frac{1}{8}d_4f_2-\frac{3}{8}d_1f_3-\frac{1}{8}d_2f_4 \\ -\frac{1}{4}d_3f_2-\frac{1}{4}d_1f_4 \\ -\frac{1}{2}d_2f_1 \\ \frac{1}{8}d_1f_1-\frac{3}{8}d_2f_2-\frac{1}{8}d_3f_3-\frac{1}{8}d_4f_4 \\ -\frac{1}{4}d_4f_1-\frac{1}{4}d_2f_3 \\ \frac{1}{8}d_3f_1-\frac{1}{8}d_4f_2-\frac{1}{8}d_1f_3-\frac{3}{8}d_2f_4 \\ -\frac{3}{8}d_3f_1-\frac{1}{8}d_4f_2-\frac{1}{8}d_1f_3+\frac{1}{8}d_2f_4 \\ -\frac{1}{4}d_3f_2-\frac{1}{4}d_1f_4 \\ -\frac{1}{8}d_1f_1-\frac{1}{8}d_2f_2-\frac{3}{8}d_3f_3+\frac{1}{8}d_4f_4 \\ -\frac{1}{2}d_3f_4 \\ -\frac{1}{4}d_4f_1-\frac{1}{4}d_2f_3 \\ -\frac{1}{8}d_3f_1-\frac{3}{8}d_4f_2+\frac{1}{8}d_1f_3-\frac{1}{8}d_2f_4 \\ -\frac{1}{2}d_4f_3 \\ -\frac{1}{8}d_1f_1-\frac{1}{8}d_2f_2+\frac{1}{8}d_3f_3-\frac{3}{8}d_4f_4 \end{pmatrix}\right)$$

Here the mathematical procedure to derive the minimal Dirac operator for a K-algebra.
First the total infinitesimal difference of the generic binary product is looked at.
$\tilde{p}$ should play the role of the searched for complex derivation, acting on a complex function $\tilde{f}$.

$d(\tilde{p}\,(\tilde{f})\otimes_{\mathbb{A}}\tilde{q}) = d(\tilde{f}\,(\tilde{q}))$

Writing this out in the generator free view by using the special case $d(g) = 0$, $d(\boldsymbol{A}) = 0$ and $d\tilde{p}=0$.
Literally, only constant K-algebras and constant metrics are presumed and $d\tilde{p}$ is extremal.

$$\boldsymbol{A}^T(p\,(f)\otimes dq) = df\,(q)$$

Using now the vector derivation dq onto both sides, is giving the Jacobi-matrix on the right side.

$$\boldsymbol{A}^T(p\,(f)\otimes E) = \frac{df(q)}{dq}$$

The Jacobi matrix can be expressed in terms of the nabla operator ∇ and vectorized, by using the Neudecker vec() function.

$\frac{df(q)}{dq} = \nabla^T \otimes f\,(q)$ and also $\mathrm{vec}(\frac{df(q)}{dq}) = (\nabla\otimes E)\,f\,(q)$

The Dirac operator P_{Dirac} is defined now as quadratic matrix, acting on the function $f\,(q)$.
Also here the equation is vectorized to be in line with the vectorized Jacobi matrix.
The same Kronecker-shift operator identity is then used as done in the derivation of the algebraic identity (one).

$p\,(f) := P_{Dirac}\,f\,(q)$ and also $\mathrm{vec}(\boldsymbol{A}^T(p\,(f)\otimes E) = \boldsymbol{A}^R p\,(f)$

Putting both vectorized sides together, the following equation to determine P_{Dirac} is available:

$$\boldsymbol{A}^R P_{Dirac}\,f\,(q) = (\nabla\otimes E)\,f\,(q)$$

This is structurally the same classical overdetermined matrix-vector problem as in the case of the algebraic one which minimal solution is:

$P^{left}_{Dirac} = \boldsymbol{A}^{RI}(\nabla\otimes E) \Leftrightarrow \boldsymbol{A}^{RI}\boldsymbol{A}^R(\nabla\otimes E)f == (\nabla\otimes E)f$

By now just using the mapping $\boldsymbol{A}\rightarrow\boldsymbol{A}^B$ the related right constraints are:

$P^{right}_{Dirac} = \boldsymbol{A}^{BRI}(\nabla\otimes E) \Leftrightarrow \boldsymbol{A}^{BRI}\boldsymbol{A}^{BR}(\nabla\otimes E)f == (\nabla\otimes E)f$

The related constraints for the existance are the generalized Cauchy-Riemann equations, now for any arbitrary constant K-algebra.

In Quantum Chromo Dynamics (QCD) the K-algebra of the eight λ matrices is as follows:

In [120]:
```
A8 = K_Algebra(8);
A_QCD = A8.A_from_multi_tab(Dict_M2A.get("QCD"));  A8.non_zero()
```

Out[120]:

$$\left[[a_{123},1],[a_{132},-1],\left[a_{147},\frac{1}{2}\right],\left[a_{156},-\frac{1}{2}\right],\left[a_{165},\frac{1}{2}\right],\left[a_{174},-\frac{1}{2}\right],[a_{213},-1],[a_{231},1],\left[a_{246},\frac{1}{2}\right],\left[a_{257},\frac{1}{2}\right],\left[a_{264},-\frac{1}{2}\right],\left[a_{275},-\frac{1}{2}\right],[a_{312},1],[a_{321},-1],\left[a_{345},\frac{1}{2}\right],\left[a_{354},-\frac{1}{2}\right],\left[a_{367},-\frac{1}{2}\right],\left[a_{376},\frac{1}{2}\right],\left[a_{417},-\frac{1}{2}\right],\left[a_{426},-\frac{1}{2}\right],\left[a_{435},-\frac{1}{2}\right],\left[a_{453},\frac{1}{2}\right],\left[a_{458},\frac{1}{2}\sqrt{3}\right],\left[a_{462},\frac{1}{2}\right],\left[a_{471},\frac{1}{2}\right],\left[a_{485},-\frac{1}{2}\sqrt{3}\right],\left[a_{516},\frac{1}{2}\right],\left[a_{527},-\frac{1}{2}\right],\left[a_{534},\frac{1}{2}\right],\left[a_{543},-\frac{1}{2}\right],\left[a_{548},-\frac{1}{2}\sqrt{3}\right],\left[a_{561},-\frac{1}{2}\right],\left[a_{572},\frac{1}{2}\right],\left[a_{584},\frac{1}{2}\sqrt{3}\right],\left[a_{615},-\frac{1}{2}\right],\left[a_{624},\frac{1}{2}\right],\left[a_{637},\frac{1}{2}\right],\left[a_{642},-\frac{1}{2}\right],\left[a_{651},\frac{1}{2}\right],\left[a_{673},-\frac{1}{2}\right]\left[a_{678},\frac{1}{2}\sqrt{3}\right],\left[a_{687},-\frac{1}{2}\sqrt{3}\right],\left[a_{714},\frac{1}{2}\right],\left[a_{725},\frac{1}{2}\right],\left[a_{736},-\frac{1}{2}\right],\left[a_{741},-\frac{1}{2}\right],\left[a_{752},-\frac{1}{2}\right],\left[a_{763},\frac{1}{2}\right],\left[a_{768},-\frac{1}{2}\sqrt{3}\right],\left[a_{786},\frac{1}{2}\sqrt{3}\right],\left[a_{845},\frac{1}{2}\sqrt{3}\right],\left[a_{854},-\frac{1}{2}\sqrt{3}\right],\left[a_{867},\frac{1}{2}\sqrt{3}\right],\left[a_{876},-\frac{1}{2}\sqrt{3}\right]\right]$$

In [121]:
```
A8.P_Dirac(),A8.P_Dirac(A8.B())  # the related minimal Dirac op's for QCD
```

Out[121]:

$$\left(\begin{pmatrix} 0 & -\frac{1}{3}d_3 & \frac{1}{3}d_2 & -\frac{1}{6}d_7 & \frac{1}{6}d_6 & -\frac{1}{6}d_5 & \frac{1}{6}d_4 & 0 \\ \frac{1}{3}d_3 & 0 & -\frac{1}{3}d_1 & -\frac{1}{6}d_6 & -\frac{1}{6}d_7 & \frac{1}{6}d_4 & \frac{1}{6}d_5 & 0 \\ -\frac{1}{3}d_2 & \frac{1}{3}d_1 & 0 & -\frac{1}{6}d_5 & \frac{1}{6}d_4 & \frac{1}{6}d_7 & -\frac{1}{6}d_6 & 0 \\ \frac{1}{6}d_7 & \frac{1}{6}d_6 & \frac{1}{6}d_5 & 0 & -\frac{1}{6}\sqrt{3}d_8-\frac{1}{6}d_3 & -\frac{1}{6}d_2 & -\frac{1}{6}d_1 & \frac{1}{6}\sqrt{3}d_5 \\ -\frac{1}{6}d_6 & \frac{1}{6}d_7 & -\frac{1}{6}d_4 & \frac{1}{6}\sqrt{3}d_8+\frac{1}{6}d_3 & 0 & \frac{1}{6}d_1 & -\frac{1}{6}d_2 & -\frac{1}{6}\sqrt{3}d_4 \\ \frac{1}{6}d_5 & -\frac{1}{6}d_4 & -\frac{1}{6}d_7 & \frac{1}{6}d_2 & -\frac{1}{6}d_1 & 0 & -\frac{1}{6}\sqrt{3}d_8+\frac{1}{6}d_3 & \frac{1}{6}\sqrt{3}d_7 \\ -\frac{1}{6}d_4 & -\frac{1}{6}d_5 & \frac{1}{6}d_6 & \frac{1}{6}d_1 & \frac{1}{6}d_2 & \frac{1}{6}\sqrt{3}d_8-\frac{1}{6}d_3 & 0 & -\frac{1}{6}\sqrt{3}d_6 \\ 0 & 0 & 0 & -\frac{1}{6}\sqrt{3}d_5 & \frac{1}{6}\sqrt{3}d_4 & -\frac{1}{6}\sqrt{3}d_7 & \frac{1}{6}\sqrt{3}d_6 & 0 \end{pmatrix}, \begin{pmatrix} 0 & \frac{1}{3}d_3 & -\frac{1}{3}d_2 & \frac{1}{6}d_7 & -\frac{1}{6}d_6 & \frac{1}{6}d_5 & -\frac{1}{6}d_4 & 0 \\ -\frac{1}{3}d_3 & 0 & \frac{1}{3}d_1 & \frac{1}{6}d_6 & \frac{1}{6}d_7 & -\frac{1}{6}d_4 & -\frac{1}{6}d_5 & 0 \\ \frac{1}{3}d_2 & -\frac{1}{3}d_1 & 0 & \frac{1}{6}d_5 & -\frac{1}{6}d_4 & -\frac{1}{6}d_7 & \frac{1}{6}d_6 & 0 \\ -\frac{1}{6}d_7 & -\frac{1}{6}d_6 & -\frac{1}{6}d_5 & 0 & \frac{1}{6}\sqrt{3}d_8+\frac{1}{6}d_3 & \frac{1}{6}d_2 & \frac{1}{6}d_1 & -\frac{1}{6}\sqrt{3}d_5 \\ \frac{1}{6}d_6 & -\frac{1}{6}d_7 & \frac{1}{6}d_4 & -\frac{1}{6}\sqrt{3}d_8-\frac{1}{6}d_3 & 0 & -\frac{1}{6}d_1 & \frac{1}{6}d_2 & \frac{1}{6}\sqrt{3}d_4 \\ -\frac{1}{6}d_5 & \frac{1}{6}d_4 & \frac{1}{6}d_7 & -\frac{1}{6}d_2 & \frac{1}{6}d_1 & 0 & \frac{1}{6}\sqrt{3}d_8-\frac{1}{6}d_3 & -\frac{1}{6}\sqrt{3}d_7 \\ \frac{1}{6}d_4 & \frac{1}{6}d_5 & -\frac{1}{6}d_6 & -\frac{1}{6}d_1 & -\frac{1}{6}d_2 & -\frac{1}{6}\sqrt{3}d_8+\frac{1}{6}d_3 & 0 & \frac{1}{6}\sqrt{3}d_6 \\ 0 & 0 & 0 & \frac{1}{6}\sqrt{3}d_5 & -\frac{1}{6}\sqrt{3}d_4 & \frac{1}{6}\sqrt{3}d_7 & -\frac{1}{6}\sqrt{3}d_6 & 0 \end{pmatrix}\right)$$

In [122]:
```
A8.P_Dirac()== -A8.P_Dirac(A8.B())
```

Out[122]: True

In [123]:
```
A8.one()
```

Out[123]: {}

In [124]:
```
A8.is_commutative()
```

Out[124]: anti-commutative

In [125]:
```
A8.is_associative()
```

Out[125]: not associative

In [126]:
```
(4*A8.Cauchy_Riemann_Eq()).list() # the related existence constraints for QCD P_Dirac()
```

Out[126]:

$$\Big[12\,d_1f_1,\ 4\,d_2f_1+8\,d_1f_2+2\,d_5f_4-2\,d_4f_5-2\,d_7f_6+2\,d_6f_7,\ 4\,d_3f_1+8\,d_1f_3-2\,d_6f_4-2\,d_7f_5+2\,d_4f_6+2\,d_5f_7,$$
$$-\sqrt{3}d_8f_6+\sqrt{3}d_6f_8+d_4f_1+d_5f_2-d_6f_3+11\,d_1f_4-d_2f_5+d_3f_6,\ -\sqrt{3}d_8f_7+\sqrt{3}d_7f_8+d_5f_1-d_4f_2-d_7f_3+d_2f_4+11\,d_1f_5+d_3f_7,$$
$$-\sqrt{3}d_8f_4+\sqrt{3}d_4f_8+d_6f_1-d_7f_2+d_4f_3-d_3f_4+11\,d_1f_6+d_2f_7,\ -\sqrt{3}d_8f_5+\sqrt{3}d_5f_8+d_7f_1+d_6f_2+d_5f_3-d_3f_5-d_2f_6+11\,d_1f_7,\ 12\,d_1f_8,$$
$$8\,d_2f_1+4\,d_1f_2-2\,d_5f_4+2\,d_4f_5+2\,d_7f_6-2\,d_6f_7,\ 12\,d_2f_2,\ 4\,d_3f_2+8\,d_2f_3+2\,d_7f_4-2\,d_6f_5+2\,d_5f_6-2\,d_4f_7,$$
$$\sqrt{3}d_8f_7-\sqrt{3}d_7f_8-d_5f_1+d_4f_2+d_7f_3+11\,d_2f_4+d_1f_5-d_3f_7,\ -\sqrt{3}d_8f_6+\sqrt{3}d_6f_8+d_4f_1+d_5f_2-d_6f_3-d_1f_4+11\,d_2f_5+d_3f_6,$$
$$-\sqrt{3}d_8f_5+\sqrt{3}d_5f_8+d_7f_1+d_6f_2+d_5f_3-d_3f_5+11\,d_2f_6-d_1f_7,\ \sqrt{3}d_8f_4-\sqrt{3}d_4f_8-d_6f_1+d_7f_2-d_4f_3+d_3f_4+d_1f_6+11\,d_2f_7,\ 12\,d_2f_8,$$
$$8\,d_3f_1+4\,d_1f_3+2\,d_6f_4+2\,d_7f_5-2\,d_4f_6-2\,d_5f_7,\ 8\,d_3f_2+4\,d_2f_3-2\,d_7f_4+2\,d_6f_5-2\,d_5f_6+2\,d_4f_7,\ 12\,d_3f_3,$$
$$-\sqrt{3}d_8f_4+\sqrt{3}d_4f_8+d_6f_1-d_7f_2+d_4f_3+11\,d_3f_4-d_1f_6+d_2f_7,\ -\sqrt{3}d_8f_5+\sqrt{3}d_5f_8+d_7f_1+d_6f_2+d_5f_3+11\,d_3f_5-d_2f_6-d_1f_7,$$
$$\sqrt{3}d_8f_6-\sqrt{3}d_6f_8-d_4f_1-d_5f_2+d_6f_3+d_1f_4+d_2f_5+11\,d_3f_6,\ \sqrt{3}d_8f_7-\sqrt{3}d_7f_8-d_5f_1+d_4f_2+d_7f_3-d_2f_4+d_1f_5+11\,d_3f_7,\ 12\,d_3f_8,$$
$$\sqrt{3}d_8f_6-\sqrt{3}d_6f_8+11\,d_4f_1-d_5f_2+d_6f_3+d_1f_4+d_2f_5-d_3f_6,\ -\sqrt{3}d_8f_7+\sqrt{3}d_7f_8+d_5f_1+11\,d_4f_2-d_7f_3+d_2f_4-d_1f_5+d_3f_7,$$
$$\sqrt{3}d_8f_4-\sqrt{3}d_4f_8-d_6f_1+d_7f_2+11\,d_4f_3+d_3f_4+d_1f_6-d_2f_7,\ 12\,d_4f_4,\ 2\,d_2f_1-2\,d_1f_2+4\,d_5f_4+8\,d_4f_5+2\,d_7f_6-2\,d_6f_7,$$
$$-2\,d_3f_1+2\,d_1f_3+d_6f_4+d_7f_5+11\,d_4f_6-d_5f_7,\ 2\,d_3f_2-2\,d_2f_3+d_7f_4-d_6f_5+d_5f_6+11\,d_4f_7,\ -\sqrt{3}d_6f_1+\sqrt{3}d_7f_2-\sqrt{3}d_4f_3+\sqrt{3}d_3f_4+\sqrt{3}d_1f_6-\sqrt{3}d_2f_7+3\,d_8f_4$$
$$+9\,d_4f_8,\ \sqrt{3}d_8f_7-\sqrt{3}d_7f_8+11\,d_5f_1+d_4f_2+d_7f_3-d_2f_4+d_1f_5-d_3f_7,\ \sqrt{3}d_8f_6-\sqrt{3}d_6f_8-d_4f_1+11\,d_5f_2+d_6f_3+d_1f_4+d_2f_5-d_3f_6,$$
$$\sqrt{3}d_8f_5-\sqrt{3}d_5f_8-d_7f_1-d_6f_2+11\,d_5f_3+d_3f_5+d_2f_6+d_1f_7,\ -2\,d_2f_1+2\,d_1f_2+8\,d_5f_4+4\,d_4f_5-2\,d_7f_6+2\,d_6f_7,\ 12\,d_5f_5,$$
$$-2\,d_3f_2+2\,d_2f_3-d_7f_4+d_6f_5+11\,d_5f_6+d_4f_7,\ -2\,d_3f_1+2\,d_1f_3+d_6f_4+d_7f_5-d_4f_6+11\,d_5f_7,\ -\sqrt{3}d_7f_1-\sqrt{3}d_6f_2-\sqrt{3}d_5f_3+\sqrt{3}d_3f_5+\sqrt{3}d_2f_6+\sqrt{3}d_1f_7+3\,d_8f_5$$
$$+9\,d_5f_8,\ \sqrt{3}d_8f_4-\sqrt{3}d_4f_8+11\,d_6f_1+d_7f_2-d_4f_3+d_3f_4+d_1f_6-d_2f_7,\ \sqrt{3}d_8f_5-\sqrt{3}d_5f_8-d_7f_1+11\,d_6f_2-d_5f_3+d_3f_5+d_2f_6+d_1f_7,$$
$$-\sqrt{3}d_8f_6+\sqrt{3}d_6f_8+d_4f_1+d_5f_2+11\,d_6f_3-d_1f_4-d_2f_5+d_3f_6,\ 2\,d_3f_1-2\,d_1f_3+11\,d_6f_4-d_7f_5+d_4f_6+d_5f_7,\ 2\,d_3f_2-2\,d_2f_3+d_7f_4+11\,d_6f_5+d_5f_6-d_4f_7,\ 12\,d_6f_6,$$
$$-2\,d_2f_1+2\,d_1f_2+2\,d_5f_4-2\,d_4f_5+4\,d_7f_6+8\,d_6f_7,\ -\sqrt{3}d_4f_1-\sqrt{3}d_5f_2+\sqrt{3}d_6f_3+\sqrt{3}d_1f_4+\sqrt{3}d_2f_5-\sqrt{3}d_3f_6+3\,d_8f_6+9\,d_6f_8,$$
$$\sqrt{3}d_8f_5-\sqrt{3}d_5f_8+11\,d_7f_1-d_6f_2-d_5f_3+d_3f_5+d_2f_6+d_1f_7,\ -\sqrt{3}d_8f_4+\sqrt{3}d_4f_8+d_6f_1+11\,d_7f_2+d_4f_3-d_3f_4-d_1f_6+d_2f_7,$$
$$-\sqrt{3}d_8f_7+\sqrt{3}d_7f_8+d_5f_1-d_4f_2+11\,d_7f_3+d_2f_4-d_1f_5+d_3f_7,\ -2\,d_3f_2+2\,d_2f_3+11\,d_7f_4+d_6f_5-d_5f_6+d_4f_7,\ 2\,d_3f_1-2\,d_1f_3-d_6f_4+11\,d_7f_5+d_4f_6+d_5f_7,$$
$$2\,d_2f_1-2\,d_1f_2-2\,d_5f_4+2\,d_4f_5+8\,d_7f_6+4\,d_6f_7,\ 12\,d_7f_7,\ -\sqrt{3}d_5f_1+\sqrt{3}d_4f_2+\sqrt{3}d_7f_3-\sqrt{3}d_2f_4+\sqrt{3}d_1f_5-\sqrt{3}d_3f_7+3\,d_8f_7+9\,d_7f_8,\ 12\,d_8f_1,\ 12\,d_8f_2,\ 12\,d_8f_3,$$
$$\sqrt{3}d_6f_1-\sqrt{3}d_7f_2+\sqrt{3}d_4f_3-\sqrt{3}d_3f_4-\sqrt{3}d_1f_6+\sqrt{3}d_2f_7+9\,d_8f_4+3\,d_4f_8,\ \sqrt{3}d_7f_1+\sqrt{3}d_6f_2+\sqrt{3}d_5f_3-\sqrt{3}d_3f_5-\sqrt{3}d_2f_6-\sqrt{3}d_1f_7+9\,d_8f_5+3\,d_5f_8,$$
$$\sqrt{3}d_4f_1+\sqrt{3}d_5f_2-\sqrt{3}d_6f_3-\sqrt{3}d_1f_4-\sqrt{3}d_2f_5+\sqrt{3}d_3f_6+9\,d_8f_6+3\,d_6f_8,\ \sqrt{3}d_5f_1-\sqrt{3}d_4f_2-\sqrt{3}d_7f_3+\sqrt{3}d_2f_4-\sqrt{3}d_1f_5+\sqrt{3}d_3f_7+9\,d_8f_7+3\,d_7f_8,\ 12\,d_8f_8\Big]$$

A.5 Other exemplary applications

A.5.1 The generalization of complex integrals

Exemplary for the complex numbers is valid:

$$\oint \tfrac{1}{z}\,dz = 2\pi i$$

For arbitrary constant K-algebras there are two chiral versions existent, here the left version (right version by $\boldsymbol{A} \to \boldsymbol{A}^B$):

$$\oint \tilde{q}^I \otimes_{\mathbb{A}} d\tilde{q} = \oint (g^T \tfrac{Hq}{q^T G q}) \otimes_{\mathbb{A}} (g^T\ dq) = g^T\ \boldsymbol{A}^T\,(H \otimes E) \oint \tfrac{q}{q^T G q} \otimes dq$$

The following procedure is examplary done for the complex numbers but can be used in similar manner for any other constant K-algebras. For the complex number, the metric is euclidean by G = E and the K-algebra is commutative $\boldsymbol{A} = \boldsymbol{A}^B$:

In [127]:
```
A2 = K_Algebra(2); A2.set_algebra(A_C);
A2.is_commutative(), A2.A == A2.B()
```

Out[127]: (commutative, True)

In [128]:
```
A2.H(),A2.G()
```

Out[128]:
$$\left(\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}, \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}\right)$$

The first term $\boldsymbol{A}^T\,(H \otimes E)$ is resulting to:

In [129]:
```
term1 = A2.A.T*(A2.H().tensor_product(A2.E())); term1
```

Out[129]:
$$\begin{pmatrix}1 & 0 & 0 & 1\\ 0 & 1 & -1 & 0\end{pmatrix}$$

Therefore the following integral over a closed curve needs to be solved here:

$$\oint \tfrac{q}{q^T q} \otimes dq$$

In [130]:
```
q = A2.number('q'); dq = A2.number('dq'); f = (q.tensor_product(dq))*(1/(q.T*q));f
```

Out[130]:
$$\begin{pmatrix} \frac{dq_1q_1}{q_1^2+q_2^2} \\ \frac{dq_2q_1}{q_1^2+q_2^2} \\ \frac{dq_1q_2}{q_1^2+q_2^2} \\ \frac{dq_2q_2}{q_1^2+q_2^2} \end{pmatrix}$$

In [131]:
```
integral(f[0].list()[0]/dq_1,q_1)
```

Out[131]:
$$\frac{1}{2}\log\left(q_1^2+q_2^2\right)$$

In [132]:
```
integral(f[1].list()[0]/dq_2,q_2)
```

Out[132]:
$$\arctan\left(\frac{q_2}{q_1}\right)$$

In [133]:
```
integral(f[2].list()[0]/dq_1,q_1)
```

Out[133]:
$$\arctan\left(\frac{q_1}{q_2}\right)$$

In [134]:
```
integral(f[3].list()[0]/dq_2,q_2)
```

Out[134]:
$$\frac{1}{2}\log\left(q_1^2+q_2^2\right)$$

The first & fourth log integrals are constant and therefore zero for the same endpoint of a closed curve.
The second integral with arctan is giving $\pi/2-(-\pi/2)=+\pi$. The third integral with arctan accordingly $-\pi/2-(\pi/2)=-\pi$.
So the resulting vector of this curved integral is:

In [135]:
```
vint = matrix([0,pi,-pi,0]);vint
```

Out[135]:
$$\begin{pmatrix} 0 & \pi & -\pi & 0 \end{pmatrix}$$

Multiplying this vector with the generator base (1,i) and the before calculated term1 is giving the expected result:

In [136]:
```
matrix([1,I])*term1*vint.T
```

Out[136]:
$$\begin{pmatrix} 2i\,\pi \end{pmatrix}$$

A.5.2 Which 2-dim K-algebras are fullfilling the four algebraic conditions of a both-handed physical vacuum?

Refer to formula system (8.3) of Algebraic Field Theory. Replace the digit '2' by the '4' or '8' in the related terms to start the calculations for 4-dim and 8-dim vacuua. For 2-dim calculation 4GB of memory is minimal required and several hours of calculation are taken into account on actual PC's. Be aware that required memory and time will grow by a factor 8 moving to 4-dim and by a factor 64 moving to 8-dim.

In [137]:
```
A2 = K_Algebra(2); A2.set_algebra(A2.ini);
```

The quadrat of the metric coefficient mc2 of a general 2-dim generator base as also the left- and right one, the hermitean operator and the condition for a both handed one are needed first.

In [138]:
```
E = A2.E();
A = A2.A;
mc2 = (A.T*A).trace()/A2.n;
o_l = A2.R().T*A2.vec(E)/mc2;
o_r = A2.R(A2.B()).T*A2.vec(E)/mc2;
H = A2.R(A2.R()).T*(o_l.tensor_product(E));
```

The four conditions (8.3):

In [139]:
```
cond1 = (H.tensor_product(E))*A-A2.B((E.tensor_product(H)*A));
cond2 = A*A.T*(A2.vec(H))-mc2*A2.vec(H);
cond3 = (A2.R(A2.B())*A2.R(A2.B()).T*A2.vec(E))-mc2*A2.vec(E);
cond4 = (A2.R()*A2.R().T*A2.vec(E))-mc2*A2.vec(E);
```

The solution constraints for the 2-dim vacuum null K-algebra A_0 is then by:

In [140]:
```
import sage.libs.ecl
sage.libs.ecl.ecl_eval("(ext:set-limit 'ext:heap-size 0)");  # reserve any memory u can get
```

In [141]:
```
result = solve(cond1.list()+cond2.list()+cond3.list()+cond4.list(),A2.A.list()); result
```

Out[141]:

$$\Bigg[a_{111}a_{112}+a_{121}a_{122}+a_{112}a_{212}+a_{122}a_{222},\ a_{111}a_{112}+a_{112}a_{122}+a_{211}a_{212}+a_{212}a_{222},\ a_{111}a_{211}+a_{211}a_{212}+a_{121}a_{221}+a_{221}a_{222},$$

$$a_{111}a_{121}+a_{121}a_{122}+a_{211}a_{221}+a_{221}a_{222},\ \frac{1}{2}a_{111}^2-\frac{1}{2}a_{112}^2+\frac{1}{2}a_{121}^2-\frac{1}{2}a_{122}^2-\frac{1}{2}a_{211}^2+a_{111}a_{212}-\frac{1}{2}a_{212}^2-\frac{1}{2}a_{221}^2+a_{121}a_{222}-\frac{1}{2}a_{222}^2,$$

$$\frac{1}{2}a_{111}^2-\frac{1}{2}a_{112}^2-\frac{1}{2}a_{121}^2+a_{111}a_{122}-\frac{1}{2}a_{122}^2+\frac{1}{2}a_{211}^2-\frac{1}{2}a_{212}^2-\frac{1}{2}a_{221}^2+a_{211}a_{222}-\frac{1}{2}a_{222}^2,$$

$$-\frac{1}{2}a_{111}^2-\frac{1}{2}a_{112}^2-\frac{1}{2}a_{121}^2-\frac{1}{2}a_{122}^2-\frac{1}{2}a_{211}^2+a_{111}a_{212}+\frac{1}{2}a_{212}^2-\frac{1}{2}a_{221}^2+a_{121}a_{222}+\frac{1}{2}a_{222}^2,$$

$$-\frac{1}{2}a_{111}^2-\frac{1}{2}a_{112}^2-\frac{1}{2}a_{121}^2+a_{111}a_{122}+\frac{1}{2}a_{122}^2-\frac{1}{2}a_{211}^2-\frac{1}{2}a_{212}^2-\frac{1}{2}a_{221}^2+a_{211}a_{222}+\frac{1}{2}a_{222}^2,$$

$$2\left(\frac{(a_{111}+a_{122})a_{111}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{a_{112}(a_{211}+a_{222})}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)a_{121}$$

$$-2\left(\frac{(a_{111}+a_{122})a_{111}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{a_{112}(a_{211}+a_{222})}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)a_{211},$$

$$2\left(\frac{(a_{111}+a_{122})a_{111}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{a_{112}(a_{211}+a_{222})}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)a_{122}$$

$$-2\left(\frac{(a_{111}+a_{122})a_{111}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{a_{112}(a_{211}+a_{222})}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)a_{212},$$

$$2\left(\frac{(a_{111}+a_{122})a_{121}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{a_{122}(a_{211}+a_{222})}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)a_{121}$$

$$-2\left(\frac{(a_{111}+a_{122})a_{121}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{a_{122}(a_{211}+a_{222})}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)a_{211},$$

$$2\left(\frac{(a_{111}+a_{122})a_{121}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{a_{122}(a_{211}+a_{222})}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)a_{122}$$

$$-2\left(\frac{(a_{111}+a_{122})a_{121}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{a_{122}(a_{211}+a_{222})}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)a_{212},$$

$$-2a_{121}\left(\frac{(a_{111}+a_{122})a_{211}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{(a_{211}+a_{222})a_{212}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)$$

$$+2\left(\frac{(a_{111}+a_{122})a_{211}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{(a_{211}+a_{222})a_{212}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)a_{211},$$

$$-2a_{122}\left(\frac{(a_{111}+a_{122})a_{211}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{(a_{211}+a_{222})a_{212}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)$$

$$+2\left(\frac{(a_{111}+a_{122})a_{211}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{(a_{211}+a_{222})a_{212}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)a_{212},$$

$$-2a_{121}\left(\frac{(a_{111}+a_{122})a_{221}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{(a_{211}+a_{222})a_{222}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)$$

$$+2a_{211}\left(\frac{(a_{111}+a_{122})a_{221}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{(a_{211}+a_{222})a_{222}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right),$$

$$-2a_{122}\left(\frac{(a_{111}+a_{122})a_{221}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{(a_{211}+a_{222})a_{222}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)$$

$$+2a_{212}\left(\frac{(a_{111}+a_{122})a_{221}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{(a_{211}+a_{222})a_{222}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right),$$

$$\left(a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2\right)\left(\frac{(a_{111}+a_{122})a_{111}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{a_{112}(a_{211}+a_{222})}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)$$

$$+2\left(a_{111}^2+a_{112}^2\right)\left(\frac{(a_{111}+a_{122})a_{111}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{a_{112}(a_{211}+a_{222})}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)$$

$$+2\left(a_{111}a_{121}+a_{112}a_{122}\right)\left(\frac{(a_{111}+a_{122})a_{121}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{a_{122}(a_{211}+a_{222})}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)$$

$$+2\left(a_{111}a_{211}+a_{112}a_{212}\right)\left(\frac{(a_{111}+a_{122})a_{211}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{(a_{211}+a_{222})a_{212}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)$$

$$+2\left(a_{111}a_{221}+a_{112}a_{222}\right)\left(\frac{(a_{111}+a_{122})a_{221}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{(a_{211}+a_{222})a_{222}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right),$$

$$2\left(a_{111}a_{121}+a_{112}a_{122}\right)\left(\frac{(a_{111}+a_{122})a_{111}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{a_{112}(a_{211}+a_{222})}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)$$

$$\left(a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2\right)\left(\frac{(a_{111}+a_{122})a_{121}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{a_{122}(a_{211}+a_{222})}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)$$

$$+2\left(a_{121}^2+a_{122}^2\right)\left(\frac{(a_{111}+a_{122})a_{121}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{a_{122}(a_{211}+a_{222})}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)$$

$$+2\left(a_{121}a_{211}+a_{122}a_{212}\right)\left(\frac{(a_{111}+a_{122})a_{211}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{(a_{211}+a_{222})a_{212}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)$$

$$+2\left(a_{121}a_{221}+a_{122}a_{222}\right)\left(\frac{(a_{111}+a_{122})a_{221}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{(a_{211}+a_{222})a_{222}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right),$$

$$2\left(a_{111}a_{211}+a_{112}a_{212}\right)\left(\frac{(a_{111}+a_{122})a_{111}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}+\frac{a_{112}(a_{211}+a_{222})}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)$$

$$+ 2\,(a_{121}a_{211} + a_{122}a_{212})\left(\frac{(a_{111}+a_{122})a_{121}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2} + \frac{a_{122}(a_{211}+a_{222})}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)$$

$$(a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2)\left(\frac{(a_{111}+a_{122})a_{211}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2} + \frac{(a_{211}+a_{222})a_{212}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)$$

$$+ 2\,(a_{211}^2+a_{212}^2)\left(\frac{(a_{111}+a_{122})a_{211}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2} + \frac{(a_{211}+a_{222})a_{212}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)$$

$$+ 2\,(a_{211}a_{221} + a_{212}a_{222})\left(\frac{(a_{111}+a_{122})a_{221}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2} + \frac{(a_{211}+a_{222})a_{222}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right),$$

$$2\,(a_{111}a_{221} + a_{112}a_{222})\left(\frac{(a_{111}+a_{122})a_{111}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2} + \frac{a_{112}(a_{211}+a_{222})}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)$$

$$+ 2\,(a_{121}a_{221} + a_{122}a_{222})\left(\frac{(a_{111}+a_{122})a_{121}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2} + \frac{a_{122}(a_{211}+a_{222})}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)$$

$$+ 2\,(a_{211}a_{221} + a_{212}a_{222})\left(\frac{(a_{111}+a_{122})a_{211}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2} + \frac{(a_{211}+a_{222})a_{212}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)$$

$$(a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2)\left(\frac{(a_{111}+a_{122})a_{221}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2} + \frac{(a_{211}+a_{222})a_{222}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)$$

$$+ 2\,(a_{221}^2+a_{222}^2)\left(\frac{(a_{111}+a_{122})a_{221}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2} + \frac{(a_{211}+a_{222})a_{222}}{a_{111}^2+a_{112}^2+a_{121}^2+a_{122}^2+a_{211}^2+a_{212}^2+a_{221}^2+a_{222}^2}\right)\Bigg]$$

Ω

```
# Python class K_Algebra() for implementing several methods required for the Algebraic Field Theory
# http://vixra.org/abs/1808.0677
# Author: Robert H. Ihde (roberthans.ihde@gmail.com)
#
# This program is free software: you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation, either version 3 of the License, or
# at any later version.
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details see <https://www.gnu.org/licenses/>.


# File K_algebra.sage (needs to be in .txt format)

# global variable for the active K-algebra (rectangular matrix)
K_algebra_active =()

# class K_Algebra() with related methods
# K_Algebra(n, *d) will set the instance of a n-dim K-algebra
# with optional related identifier (string) d for the field variable (default='a').
# Methods:
# ().n -> the dimension of the active K-algebra
# ().ini -> the arbitrary initial representation of the K-algebra
# ().A -> the current active K-algebra
# ().set_algebra(A) -> load a specific K-algebra A, expects a (n^2,n) matrix
# ().var_i1(d,*m) -> create an indexed variable labeled d, opt. other dimension= (int) m
# ().var_i2(d) -> create a tuple indexed variable labeled d
# ().var_i3(d) -> create a triple indexed variable labeled d
# ().number(d) -> create a number for the actual instance labeled d
# ().qmatrix(d) -> create a quadratic matrix from a n^2-dim vector labeled d
# ().Amatrix(d) -> create another (n^2,n) rectangular matrix labeled d
# ().multiply(p,q) -> algebraic multiplication of two numbers p,q (of type ().number)
# ().E(*m) -> identity matrix of dim n (can be set optionally to dim m)
# ().u(k) -> unit vector of dim n in the k-th direction
# ().T() -> short cut alias of the transpose() function
# ().I() -> short cut alias of the pseudoinverse() function
# ().vec(*M) -> Neudecker vec function of .A or if M given for that matrix
# ().B() -> the bilateral shifted matrix .A in the first two indices of the triple index
# ().is_commutative() -> check for commutativity of the active K-algebra
# ().is_associative() -> check for associativity of the active K-algebra
# ().R() -> the right shifted matrix .A in the triple indices
# ().v2M() -> helping function to output the multiplication tab (used in multi_tab)
# ().multi_tab(*A) -> outputs the multiplication table for .A or for a related A given matrix
# ().A_from_multi_tab(M) -> creates the rectangular matrix A from the multiplication table M
# ().type_of_one() -> determines the chirality of a one (right, left, both) or returns {} if not exists
# ().one() -> returns the one in case exists, returns {} if not exists
# ().A_gauge() -> the gauging factor of the algebraic generator base
# ().detect_type(p) -> returns ('vect','qmat','rmat')  in case of p is of (vector,nxn matrix,nxm matrix)
# ().H(*q) -> minimal dual generator operator or acting on a given number q, returns {} if not exists
# ().G(*q) -> minimal dual metric field operator or acting on a given number q, returns {} if not exist
# ().B_cartesianXY(n,m) -> helping B operator for dim (n,m) used in method XY_cartesian()
# ().XY_cartesian(Y) ->  cartesian product between the active K-algebra X and given K-algebra Y
# ().P_Dirac() -> minimal Dirac operator for the active K-algebra
# ().Cauchy_Riemann_Eq() -> the Cauchy-Riemann constraints for the active K-algebra
# ().index3_2(j,k,l) -> reformats the triple index to a 2-dim matrix standard index (start index=1)
# ().index2_3(j,k) -> reformats a 2-dim matrix standard index to a related triple index (start index=1)
# ().H_R(*q) -> regular dual generator operator or acting on given number q, returns {} if not exists
# ().G_R(*q) -> regular dual metric field operator or acting on given number q, returns {} if not exist
# ().non_zero() -> returns the non zero entries of the active rectangular matrix

class K_Algebra():

    def __init__(self,n,*d):
        global K_algebra_active
        K_algebra_active = ()
        if d == ():
            d = 'a'
        self.n = n
        self.ini = matrix(SR,n*n, n,\
                 var(','.join([str(d)+'_%d%d%d' % (j,k,l) for j in [1..n] for k in [1..n] for l in [1..n]])))
        self.A = K_algebra_active
        if self.A == ():
            self.A = self.ini
            K_algebra_active = self.ini

    def set_algebra(self,X):
        global K_algebra_active
        K_algebra_active = ()
        self.A = X
        K_algebra_active = X
```

```
    def var_i1(self,d, *s):
        if s==():
            m =self.n
        else:
            m=s[0]
        j = var('j')
        return var(','.join([str(d)+'_%d' % (j) for j in [1..m]]))

    def var_i2(self,d):
        (j,k) = var('j','k')
        return var(','.join([str(d)+'_%d%d' % (j,k) for j in [1..self.n] for k in [1..self.n]]))

    def var_i3(self,d):
        (j,k,l) = var('j','k','l')
        return var(','.join([str(d)+'_%d%d%d' % (j,k,l) for j in [1..self.n] for k in [1..self.n] for l in
[1..self.n]]))

    def number(self,d):
        if isinstance(d, basestring):
            return matrix(SR, self.n, 1, self.var_i1(d))
        else:
            return matrix(d).transpose()

    def qmatrix(self,d):
        return matrix(SR,self.n, self.n, self.var_i2(d))

    def Amatrix(self,d):
        return matrix(SR,self.n^2,self.n,self.var_i3(d))

    def multiply(self,p,q):
        return (self.A).transpose()*(p.tensor_product(q))

    def E(self, *s):
        if s == ():
            return identity_matrix(self.n)
        else:
            return identity_matrix(s[0])

    def u(self,k):
        return identity_matrix(self.n)[:,k-1]

    def T(self, *s):
        if s==():
            return (self.A).transpose()
        else:
            return (s[0].transpose())

    def I(self,*s):
        if s == ():
            return (self.A).pseudoinverse()
        else:
            return (s[0]).pseudoinverse()

    def vec(self,*s):
        if s == ():
            return matrix([(self.A).transpose().list()]).transpose()
        return matrix([(s[0]).transpose().list()]).transpose()

    def B(self, *s):
        j=var('j')
        if s == ():
            Y = self.A
        else:
            Y = s[0]
        X=(sum([((self.u(j).transpose()).tensor_product(identity_matrix(self.n))).tensor_product(self.u(j))
for j in [1..self.n ]]))*Y
        return X

    def is_commutative(self):
        if self.A == self.B():
            return "commutative"
        if self.A == -self.B():
            return "anti-commutative"
        return "not commutative"

    def is_associative(self):
        if ((self.A).tensor_product(identity_matrix(self.n)))*self.A ==
(identity_matrix(self.n)).tensor_product(self.A)*self.A:
            return 'associative'
        return 'not associative'
```

```
    def R(self, *s):
        j=var('j')
        if s == ():
            Y = self.A
        else:
            Y = s[0]
        X =
(sum([(self.u(j).tensor_product(identity_matrix(self.n)))*(Y.transpose())*((identity_matrix(self.n)).tensor_p
roduct(self.u(j))) for j in [1..self.n ]]))
        return X

    def v2M(self,q):
        j= var('j')
        n=self.n
        return
sum([((self.u(j)).transpose().tensor_product(identity_matrix(self.n)))*q.tensor_product((self.u(j)).transpose
()) for j in [1..self.n]]).transpose()

    def multi_tab(self, *s):
        if s == ():
            return self.v2M(self.A*self.number('g'))
        else:
            return self.v2M(s[0]*self.number('g'))

    def A_from_multi_tab(self,G):
        global K_algebra_active
        K_algebra_active = ()
        n = self.n
        H = self.multi_tab()
        eq = self.vec(self.multi_tab(self.Amatrix('a'))-G)
        rs=[eq.list()[j].coefficient(self.var_i1('g')[k]) for j in range(0,n^2) for k in range(0,n)]
        rt=flatten(solve(rs,self.var_i3('a')))
        X = matrix([[SR(str(rt[(j-1)+(k-1)*n]).split()[2]) for j in [1..n]] for k in [1..n^2]])
        self.A = X
        K_algebra_active = X
        return X

    def type_of_one(self):
        tst_l = expand(simplify((self.R()*self.I(self.R()))))
        tst_e = expand(simplify(self.vec(self.E())))
        tst_r = expand(simplify(self.R(self.B()) * self.I(self.R(self.B()))))
        check = 0
        if tst_l * tst_e == tst_e:
            check = 1
        if tst_r * tst_e == tst_e:
            check = check+2
        if check == 1:
            return 'left-handed one'
        if check == 2:
            return 'right-handed one'
        if check == 3:
            return 'both-handed one'
        return 'nonexistent one'

    def one(self):
        if self.type_of_one() == 'both-handed one' or self.type_of_one() == 'left-handed one':
            return expand(simplify(self.I(self.R()) * self.vec(self.E())))
        if self.type_of_one() == 'right-handed one':
            return expand(simplify(self.I(self.R(self.B())) * self.vec(self.E())))
        return {}

    def A_gauge (self):
        X = ((((self.A).transpose()*self.A).trace())/((self.E()).trace())).sqrt()
        self.A_gauge = X
        return X

    def detect_type(self,s):
        if s.ncols() == 1:
            return "vect"
        if s.ncols() == s.nrows():
            return "qmat"
        if s.ncols() != s.nrows():
            return "rmat"
```

```
    def H(self, *s):
        o = self.one()
        if o =={}:
            return {}
        mc2 = (self.A_gauge())^2
        X = self.T(self.R(self.R()))*o.tensor_product(self.E())
        if X != X.transpose():
            return {}
        if X*X !=self.E():
            return {}
        tst_r = expand(simplify((mc2*self.vec(X))))
        tst_l = expand(simplify(self.A*self.T(self.A)*self.vec(X)))
        if tst_l == tst_r:
            if X != () and s != ():
                if self.detect_type(s[0]) == 'vect':
                    return X*s[0]
                if self.detect_type(s[0]) == 'qmat':
                    return X*s[0]*X
                if self.detect_type(s[0]) == 'rmat':
                    return (X.tensor_product(X))*s[0]*X
            if X != () and s == ():
                return X
        return {}

    def G(self, *s):
        o= self.one()
        H= self.H()
        if  o == {}:
            return {}
        if  H == {}:
            return {}
        X = self.T(self.R(self.R()))*self.T(self.I(o)).tensor_product(H)
        if X != X.transpose():
            return {}
        if s == ():
            return X
        if X != () and s != ():
            if self.detect_type(s[0]) == 'vect':
                return X*s[0]
            if self.detect_type(s[0]) == 'qmat':
                return X*s[0]*X
            if self.detect_type(s[0]) == 'rmat':
                return (X.tensor_product(X))*s[0]*X
        return {}

    def B_cartesian(self,m):
        j= var('j')
        n= self.n
        X = sum([ (identity_matrix(m)[:,j-
1].tensor_product(self.E(n))).tensor_product(self.T(identity_matrix(m)[:,j-1])) for j in [1..m]])
        return X

    def XY_cartesian(self,Y):
        n = self.n
        m = min(Y.ncols(),Y.nrows())
        X =
(((self.E(n)).tensor_product(self.B_cartesian(m))).tensor_product(self.E(m)))*((self.A).tensor_product(Y))
        return X

    def P_Dirac(self,*s):
        if s == ():
            return self.I(self.R())*(self.number('d')).tensor_product(self.E())
        else:
            return self.I(self.R(s[0]))*(self.number('d')).tensor_product(self.E())

    def Cauchy_Riemann_Eq(self,*s):
        mc2 = self.A_gauge()^2
        if s == ():
            return mc2*(self.E((self.n)^2)-
self.R()*self.I(self.R()))*(self.number('d').tensor_product(self.E())*self.number('f'))
        else:
            return mc2*(self.E((self.n)^2)-
self.R(s[0])*self.I(self.R(s[0])))*(self.number('d').tensor_product(self.E())*self.number('f'))

    def index3_2(self,j,k,l):
        n= self.n
        return (j + (n - 1)*(j - 1) + k - 1, l);

    def index2_3(self,j,k):
        n = self.n
        return (ceil(j/n), j - ceil(j/n)*n + n, k)
```

```
def H_R(self, *s):
    o = self.one()
    if o =={}:
        return {}
    n = self.n-1
    q = self.number('q')
    M = (self.A.T*(self.E().tensor_product(q)))
    Z = (M^-1)*self.one()
    X = matrix([[numerator(Z.list()[k]).coefficient(q[j][0]) for j in [0..n] ] for k in [0..n] ])
    if X != X.transpose():
        return {}
    if X*X != self.E():
        return {}
    if X != () and s != ():
        if self.detect_type(s[0]) == 'vect':
            return X*s[0]
        if self.detect_type(s[0]) == 'qmat':
            return X*s[0]*X
        if self.detect_type(s[0]) == 'rmat':
            return (X.tensor_product(X))*s[0]*X
    if X != () and s == ():
        return X
    return {}

def G_R(self, *s):
    o= self.one()
    H= self.H_R()
    if  o == {}:
        return {}
    if  H == {}:
        return {}
    X = self.T(self.R(self.R()))*self.T(self.I(o)).tensor_product(H)
    if X != X.transpose():
        return {}
    if s == ():
        return X
    if X != () and s != ():
        if self.detect_type(s[0]) == 'vect':
            return X*s[0]
        if self.detect_type(s[0]) == 'qmat':
            return X*s[0]*X
        if self.detect_type(s[0]) == 'rmat':
            return (X.tensor_product(X))*s[0]*X
    return {}

def non_zero(self):
    var('i')
    return [[self.ini.list()[i]]+[(self.A).list()[i]] for i, e in enumerate((self.A).list()) if e != 0]
```

```
# Dictionary Dict_M2A() of multiplication tables for usual K-algebra up to dim=8

#    "C"     the 2-dim classical complex numbers where i*i = -1
#    "DC"    the 2-dim dual complex numbers where i*i = 0
#    "SC"    the 2-dim split-complex numbers where i*i = +1
#    "E2"    the 2-dim "euclidean numbers", defined by two stacked identity matrices
#    "OM2"   the 2-dim "one-numbers", defined by a rectangular matrix only with one's
#    "AND"   the 2-dim "and-numbers", defined by the logical AND operation
#    "OR"    the 2-dim "or-numbers", defined by the logical OR operation
#    "VP3"   the 3-dim vector cross product
#    "E3"    the 3-dim "euclidean numbers", defined by three stacked identity matrices
#    "H"     the 4-dim Hamilton quaternions
#    "PSpin" the 4-dim Pauly spin matrices including the identity
#    "D4"    the 4-dim K-algebra of the gamma matrices from the Dirac equation of the electron
#    "E4"    the 4-dim "euclidean numbers", defined by four stacked identity matrices
#    "E5"    the 5-dim "euclidean numbers", defined by five stacked identity matrices
#    "E6"    the 6-dim "euclidean numbers", defined by six stacked identity matrices
#    "E7"    the 7-dim "euclidean numbers", defined by seven stacked identity matrices
#    "E8"    the 8-dim "euclidean numbers", defined by eight stacked identity matrices
#    "QCD"   the 8-dim Gell-Mann lambda matrices in quantum chromo dynamics
#    "O"     the 8-dim Cayley octonions


var(','.join(['g'+'_%d' % (j) for j in [1..8]]));
Dict_M2A = {};
Dict_M2A.update({"C":matrix(SR,2,2,[[g_1,g_2],[g_2,-g_1]])});
Dict_M2A.update({"DC":matrix(SR,2,2,[[g_1,g_2],[g_2,0]])});
Dict_M2A.update({"SC":matrix(SR,2,2,[[g_1,g_2],[g_2,+g_1]])});
Dict_M2A.update({"E2":matrix(SR,2,2,[[g_1,g_2],[g_1,g_2]])});
Dict_M2A.update({"OM2":matrix(SR,2,2,[[g_1+g_2,g_1+g_2],[g_1+g_2,g_1+g_2]])});
Dict_M2A.update({"AND":matrix(SR,2,2,[[g_1,g_1],[g_1,g_2]])});
Dict_M2A.update({"OR":matrix(SR,2,2,[[g_2,g_2],[g_2,g_1]])});
Dict_M2A.update({"VP3":matrix(SR,3,3,[[0,g_3,-g_2],[-g_3,0,g_1],[g_2,-g_1,0]])});
Dict_M2A.update({"E3":matrix(SR,3,3,[[g_1,g_2,g_3],[g_1,g_2,g_3],[g_1,g_2,g_3]])});
Dict_M2A.update({"H":matrix(SR,4,4,[[g_1,g_2,g_3,g_4],[g_2,-g_1,g_4,-g_3],[g_3,-g_4,-g_1,g_2],[g_4,g_3,-g_2,-
g_1]])});
Dict_M2A.update({"PSpin":matrix(SR,4,4,[[g_1,g_2,g_3,g_4],[g_2,g_1,i*g_4,-i*g_3],
                                        [g_3,-i*g_4,g_1,i*g_2],[g_4,i*g_3,-i*g_2,g_1]])});
Dict_M2A.update({"D4":matrix(SR,4,4,[[g_1,g_4,-i*g_4,g_3],[g_2,g_3,i*g_3,-g_4],
                                     [-g_3,-g_2,i*g_2,-g_1],[-g_4,-g_1,-i*g_1,g_2]])*(-i/2)});
Dict_M2A.update({"E4":matrix(SR,4,4,[[g_1,g_2,g_3,g_4],[g_1,g_2,g_3,g_4],[g_1,g_2,g_3,g_4],[g_1,g_2,g_3,g_4]]
)});
Dict_M2A.update({"E5":matrix(SR,5,5,[[g_1,g_2,g_3,g_4,g_5],[g_1,g_2,g_3,g_4,g_5],[g_1,g_2,g_3,g_4,g_5],
                                     [g_1,g_2,g_3,g_4,g_5],[g_1,g_2,g_3,g_4,g_5]])});
Dict_M2A.update({"E6":matrix(SR,6,6,[[g_1,g_2,g_3,g_4,g_5,g_6],[g_1,g_2,g_3,g_4,g_5,g_6],[g_1,g_2,g_3,g_4,g_5
,g_6],

[g_1,g_2,g_3,g_4,g_5,g_6],[g_1,g_2,g_3,g_4,g_5,g_6],[g_1,g_2,g_3,g_4,g_5,g_6]])});
Dict_M2A.update({"E7":matrix(SR,7,7,[[g_1,g_2,g_3,g_4,g_5,g_6,g_7],[g_1,g_2,g_3,g_4,g_5,g_6,g_7],
                                     [g_1,g_2,g_3,g_4,g_5,g_6,g_7],[g_1,g_2,g_3,g_4,g_5,g_6,g_7],
                                     [g_1,g_2,g_3,g_4,g_5,g_6,g_7],[g_1,g_2,g_3,g_4,g_5,g_6,g_7],
                                     [g_1,g_2,g_3,g_4,g_5,g_6,g_7]])});
Dict_M2A.update({"E8":matrix(SR,8,8,[[g_1,g_2,g_3,g_4,g_5,g_6,g_7,g_8],[g_1,g_2,g_3,g_4,g_5,g_6,g_7,g_8],
                                     [g_1,g_2,g_3,g_4,g_5,g_6,g_7,g_8],[g_1,g_2,g_3,g_4,g_5,g_6,g_7,g_8],
                                     [g_1,g_2,g_3,g_4,g_5,g_6,g_7,g_8],[g_1,g_2,g_3,g_4,g_5,g_6,g_7,g_8],
                                     [g_1,g_2,g_3,g_4,g_5,g_6,g_7,g_8],[g_1,g_2,g_3,g_4,g_5,g_6,g_7,g_8]])});
Dict_M2A.update({"QCD":matrix(SR,8,8,[[0,g_3,-g_2,1/2*g_7,-1/2*g_6,1/2*g_5,-1/2*g_4,0],
                                      [-g_3,0,g_1,1/2*g_6,1/2*g_7,-1/2*g_4,-1/2*g_5,0],
                                      [g_2,-g_1,0,1/2*g_5,-1/2*g_4,-1/2*g_7,1/2*g_6,0],
                                      [-1/2*g_7,-1/2*g_6,-1/2*g_5,0,1/2*sqrt(3)*g_8+1/2*g_3,1/2*g_2,1/2*g_1,-
1/2*sqrt(3)*g_5],
                                      [1/2*g_6,-1/2*g_7,1/2*g_4,-1/2*sqrt(3)*g_8-1/2*g_3,0,-
1/2*g_1,1/2*g_2,1/2*sqrt(3)*g_4],
                                      [-1/2*g_5,1/2*g_4,1/2*g_7,-1/2*g_2,1/2*g_1,0,1/2*sqrt(3)*g_8-1/2*g_3,-
1/2*sqrt(3)*g_7],
                                      [1/2*g_4,1/2*g_5,-1/2*g_6,-1/2*g_1,-1/2*g_2,-
1/2*sqrt(3)*g_8+1/2*g_3,0,1/2*sqrt(3)*g_6],
                                      [0,0,0,1/2*sqrt(3)*g_5,-1/2*sqrt(3)*g_4,1/2*sqrt(3)*g_7,-
1/2*sqrt(3)*g_6,0]])});
Dict_M2A.update({"O":matrix(SR,8,8,[[g_1,g_2,g_3,g_4,g_5,g_6,g_7,g_8],
        [g_2,-g_1,g_4,-g_3,g_6,-g_5,-g_8,g_7],
        [g_3,-g_4,-g_1,g_2,g_7,g_8,-g_5,-g_6],
        [g_4,g_3,-g_2,-g_1,g_8,-g_7,g_6,-g_5],
        [g_5,-g_6,-g_7,-g_8,-g_1,g_2,g_3,g_4],
        [g_6,g_5,-g_8,g_7,-g_2,-g_1,-g_4,g_3],
        [g_7,g_8,g_5,-g_6,-g_3,g_4,-g_1,-g_2],
        [g_8,-g_7,g_6,g_5,-g_4,-g_3,g_2,-g_1]])});

# Dictionary for rectangular matrices, created from a function
Dict_A2A = {};
Dict_A2A.update({"N2":matrix(QQ, 4,2, lambda i, j: 2*i+j+1)});
Dict_A2A.update({"NI2":matrix(QQ, 4,2, lambda i, j: 1/(2*i+j+1))});
Dict_A2A.update({"R2_Z":random_matrix(ZZ,4,2)});
Dict_A2A.update({"R2_Q":random_matrix(QQ,4,2)});
Dict_A2A.update({"R2_R":random_matrix(RR,4,2)});

# infix operator for the algebraic multiplication p *A* q
@infix_operator('multiply')
def A(p,q):
    return K_algebra_active.transpose()*(p.tensor_product(q))
```