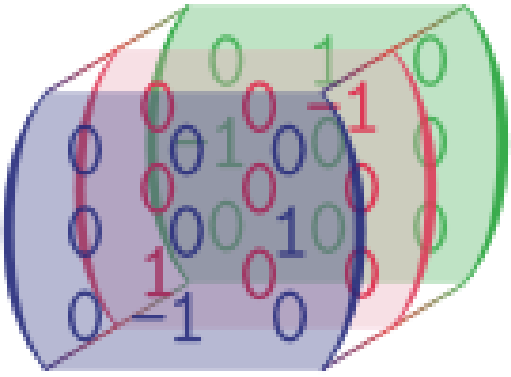
Exploring Math $\Sigma_{math}$ with EIGENMATH

# Some Tapas of Tensors

**From KRONECKER delta to the CHRISTOFFEL tensor**

$$\epsilon_{ijk} =$$



**Dr. Wolfgang Lindner**
dr.w.g.Lindner@gmail.com
Leichlingen, Germany
2023

# Contents

## Preface

Some *tapas* of tensors: this booklet would like to whet your appetite to immerse yourself into the world of tensors with small bites of special tensors to take a closer look at tensor calculus. For the novice, tensor notations and operations are somehow clumsy and uncomfortable and accompanied by heavy calculations. Therefore here is no forbidding theory presented but instead the focus is on praxis with selected examples - using EIGENMATH as your computer algebraic companion to unburden calculation in this field. We only cite some of the necessary underlying mathematical definitions and facts to be able to show the corresponding implementation of the concepts into the language of CAS EIGENMATH.

For the theory of Tensor Analysis I strongly recommend the fantastic and crystal-clear treatment by Pavel GRINFELD. His book [1] focusses on comprehension and illuminates mathematical concepts by well choosen worked examples and 354 exercises. It is a must to read. No other text I am aware of can compete with his presentation.

Looking back at my first contact with EIGENMATH I was very impressed by the ingenious and sophisticated treatment of the RIEMANN tensor in the calculation of the Schwarzschild metric by George Weigt in [15]. This script is an eye opener and a convincing powerful example of the amazing efficiency of EIGENMATH.

The collection of 25 short EIGENMATH example scripts and 27 exercises in this booklet to cope with tensor operations not only want to help the reader to dive into the theory of tensors e.g. along GRINFELD's book, but also to become comfortable with the use of the CAS EIGENMATH in this field. It aims to be a source of help and inspiration for another round of thoughtful activity with respect to tensors - supplementary to paper & pencil calculations, now from the perspective of a compact CAS. So have fun on our tour from the KRONECKER delta via the omnipresent metric tensor to the CHRISTOFFEL symbol. The forthcoming booklet on Elementary Differential Geometry will e.g. address the RIEMANN tensor $R$ and the curvature tensor $B$ to calculate the GAUSS resp. the mean curvature of surfaces in $\mathbb{R}^3$.

For the inspection or running an EIGENMATH script no installation is necessary, *everything runs directly online*: a click on a link in this text is enough to invoke the corresponding script - and by a click on the `RUN` button the calculation is made, allowing further free inputs form the user. If you own a Mac or Linux PC, there is the option to install the app EIGENMATH free of charge and run the scripts by *mark–copy–paste* into the EIGENMATH window.
I want to thank George WEIGT for his friendly support with tips and hints while writing these notes.

Wolfgang Lindner
Leichlingen, Germany, January 2023

# 1 What is a tensor?

We do not give a theoretical definition what a tensor mathematically is. Instead we visit and explore some specimens and individuals in the zoo of tensors.
Remember, that we denote a vector $v = (a_1, ...a_2) \in \mathbb{R}^n$ as a list of entries, e.g. $v = (3, 2) \in \mathbb{R}^2$. In the language of tensors we will denote $v$ often compact as $a_i$, meaning the whole list *or* the individual entry at index $i$, i.e. $a_i = (a_1, ...a_2)$, but $a_1 = 3$
We denote a matrix $M = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \in \mathbb{R}^{2,2}$ as a table of entries, e.g. $M = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \in \mathbb{R}^{2,2}$. In the language of tensors we will denote $M$ often as $a_{ij}$, meaning the whole table *or* the individual entry at the index pair $(i, j)$, i.e. $a_{ij} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$, but $a_{21} = M[2, 1] = 3$.
To sum up:

| LEXICON | *Math* | EIGENMATH |
|---|---|---|
| *vector* $a = (a_1, a_2)$ in $\mathbb{R}^2$ | $a_i$ | `a=(a1, a2)` |
| *matrix* $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ | $a_{ij}$ | `M=((a11,a12),(a21,a22))` |
| *tensor* T | $T_{ijk}$ | `Tijk` |

In a figurative sense, tensors are 'multidimensional' matrices.

**Example 1.** We define and test a tensor $T$, cf. the example by GRINFELD [1], page 94.

```
# EIGENMATH
i = quote(i)                    -- clear i as global defined imaginary unit ..

-- define/calculate tensor elements of T as function values

Tijk(i,j,k) = i^2*j*k           -- .. otherwise this term would be -jk
Tijk                            --(1)
Tijk(1,2,1)                     --(2)  1,2,1 = 1st matrix, 2nd row, 1st entry

-- tensor T as multidimensional 'matrix'

Tijk = zero(2, 2,2)             -- empty container for tensor T
for(i,1,2,                      -- three loops to fill container Tijk
  for(j,1,2,
    for(k,1,2,  Tijk[i,j,k] = i^2*j*k)))
Tijk                            -- (3)
Tijk[1,2,1]                     -- (4) 1st matrix, 2nd row, 1st entry
```

• Mark & Copy the blue code lines. ▷ Then Click here and Paste it into the input box □RUN. Press RUN.[1]
Anyway: ▷ Click here to run the script.

[1] Do not forget to *click into the Online form* to give it the focus. You have the focus, if the EIGENMATH Online frame change to blue. Please check, wether all input lines are pasted with the right NEW LINE ending! Otherwise correct the pasting online.

EIGENMATH output:

$$T_{ijk} = i^2 j\, k$$

$$2$$

$$T_{ijk} = \begin{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \\ \begin{bmatrix} 4 & 8 \\ 8 & 16 \end{bmatrix} \end{bmatrix}$$

(1): tensor T defined as function $T_{ijk} := i^2 jk$.
Figure 1: (2): value of tensor T at index triple $(i, j, k) = (1, 2, 1)$.
(3): tensor T as multidimensional 'matrix' $[(::), (::)]$.

**Comment.** We define tensor $T$ in two ways. First $T$ is defined by computing each individual tensor value by the term $i^2jk$ in (1). Here you have to address the value via `Tijk(..)` for the function call. Alternatively $T$ is defined in (2) as a multidimensional matrix whose values are read in with matrix-access brackets `Tijk[..]`. In this way the complete table $T$ of tensor values is available at once.

**Exercise 1.** (Tensor $T_{ijkl}$ with prescribed values.)

$$T_{ijkl} = \begin{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} & \begin{bmatrix} 2 & 4 \\ 4 & 8 \end{bmatrix} \\ \begin{bmatrix} 4 & 8 \\ 8 & 16 \end{bmatrix} & \begin{bmatrix} 8 & 16 \\ 16 & 32 \end{bmatrix} \end{bmatrix}$$

Determine the function and the table-oriented formula for the given tensor $T$.
Do access for the value 16 in the 3th matrix inside tensor $T$.

*Solution* N$^o$**1**: We give the defining relation as function value.

```
Tijkl(i,j,k,l) = i^2*j*k*l   -- defining function term
Tijkl(i,j,k,l)               -- the tensor access formula
Tijkl(2,1,2,2)               -- grasp value at (2,1,2,2)
```

*Solution* N$^o$**2**: We define $T$ as a tensor 'table'.

```
Tijkl = zero(2, 2,2,2)  -- empty container for tensor Tijkl
Tijkl
```

```
for(i,1,2, for(j,1,2,
           for(k,1,2,
           for(l,1,2, Tijkl[i,j,k,l] = i^2*j*k*l))))

Tijkl                   -- the tensor 'table' T
Tijkl[2,1,2,2]          -- grasp value at (2,1,2,2)
```

• Mark & Copy the blue code lines. ▷ Then Click here and Paste it into the input box □RUN. Press RUN.
Or ▷ Click here to run the script.
You should see the tensor $T$ as shown in the beginning of the exercise.

**Remark.** The outputs of the tensors $T_{ijk}$ and $T_{ijkl}$ above demonstrate, that EIGENMATH has a marvelous automatic build-in mechanism to format multidimensional objects in a clear way, that allows easy inspections. No special format options are necessary.

# 2 The KRONECKER delta tensor

In mathematics, the KRONECKER delta (named after Leopold KRONECKER) is a function of two variables ('indices'), usually non-negative integers. The function has value 1 if the inputs are equal, and 0 otherwise.

## 2.1 Definition

The KRONECKER delta is defined by

$$\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j, \\ 1 & \text{if } i = j. \end{cases}$$

We implement two versions of the KRONECKER $\delta$ tensor in EIGENMATH.

| LEXICON | *Math* | EIGENMATH |
|---|---|---|
| KRONECKER $\delta$ function | $\delta(i,j)$ | `delta(i,j)` |
| KRONECKER $\delta$ tensor | $\delta_{ij}$ | `deltaij[i,j]` |

```
# EIGENMATH

-- Define KRONECKER delta function
delta(i,j) = test( i=j, 1, 0)       -- (1)

delta11 = delta(1,1)                -- (2)
delta11                             -- (3)

delta10 = delta(1,0)
delta10

-- Define KRONECKER delta 2-tensor
deltaij = zero(2,2)                 -- (4)
   deltaij[1,1] = 1
   deltaij[2,2] = 1

deltaij                             -- the whole tensor
deltaij[1,2]                        -- (5), watch the [.] brackets
```

**Comment.** Definition (1) is a direct formulation of the mathematical definition. In (2) we test the function `delta` on the input pair $(1,1)$ and catch the result in the identifier `delta11`. Its return value in (3) is automatically written in pretty-print *LaTeX* shape $\delta_{ij}$ in the EIGENMATH-output region - which has nothing to do with the tensorial definition in (4), which is formatted by EIGENMATH also as $\delta_{ij}$ and therefore looks like (3).

• ▷ Click here to run the script.
EIGENMATH output:

$$\delta_{11} = 1$$

$$\delta_{10} = 0$$

$$\delta_{ij} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$0$$

**Exercise 2.** Argue, whether it is possible to define the KRONECKER delta tensor using the self-written function 'signum' `sign`. Do some tests in EIGENMATH's script window.

```
sign(x) = test(x>0, 1, -1)
deltaij(i,j) = 1 - abs(sign(i-j))
```

## 2.2 Applications

### 2.2.1 The inner product of vectors

The inner product of vectors can be written as

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i,j=1}^{n} a_i \delta_{ij} b_j = \sum_{i=1}^{n} a_i b_i$$

where i and j take the values $1, 2, ..., n$ and the vectors $\mathbf{a}$ and $\mathbf{b}$ are defined as arbitrary n-tuples $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{b} = (b_1, b_2, ..., b_n)$.
We see how in the above equation the values of the KRONECKER delta reduce the double summation over $i$ and $j$ to a single summation over $i$ only. This is a first indication of the '*contract*'ing effect of $\delta_{ij}$.
In the following demo it is comfortable to use the function definition of $\delta_{ij}$.

```
# EIGENMATH

delta(i,j) = test( i=j, 1, 0)

ip(a,b) = sum(i,1,dim(a),    -- inner product via double sum
          sum(j,1,dim(b),    a[i]*delta(i,j)*b[j]))   -- (1)

a=(1,2,3)
b=(3,2,1)

ip(a,b)           -- (2)
dot(a,b)          -- (3)
```

EIGENMATH output: both calculations using function `ip(.)` in (2) or the build-in dot-product `dot(.)` in (3) give the same value 10.
• ▷ Click here to run the script.

### 2.2.2 The Einstein summation convention

Sometimes the Kronecker delta $\delta_{ij}$ is written with one index as superscript and the other as subscript, i.e. $\delta_{ij} = \delta^i_j$. If one now writes $\delta_{ii} = \delta^i_i$, this is interpreted in tensor language in the sense of EINSTEIN as a hidden summation behind the scene in the ambient space, e.g. in $\mathbb{R}^3$ to suppress the summation sign $\sum$ in tensor expressions.

▷ (EINSTEIN summation convention) *A summation in a tensor expression is implicit done, when an index appears twice, once as a subscript ('lower index') and once as a superscript ('upper index').*

Ergo, summation symbols can be 'eliminated or forgotten' by using Einstein's convention, giving very compact tensorial notations - but with the drawback to decode the terms in your mind. The application of the Einstein-summation-convention is often called a *contract*ion:

$$\delta^i_i = \delta_{ii} \equiv \sum_{i=1}^{3} \delta_{ii} \qquad \text{in } \mathbb{R}^3$$

$$\downarrow \textit{ contract}$$

$$\delta^i_i$$

$$\downarrow \textit{ i.e. sum}$$

$$3$$

Here is the EINSTEIN summation convention applied in EIGENMATH:

```
# EIGENMATH
-- EINSTEIN summation convention
delta(i,j) = test( i=j, 1, 0)

deltaii = sum(i,1,3, delta(i,i))  -- Einstein's implicit sum in R^3
deltaii
```

• Mark & Copy the blue code lines. ▷ Then Click here and Paste it into the input box ☐RUN. Press RUN.
EIGENMATH output: $\delta_{ij} = 3$

**Remark.** The dot product $\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i$ of two vectors in 2.2.2 can be thought of as a contraction a la EINSTEIN's convention. The vectors are first multiplied element-wise and then contracted, i.e. 'summed up'. Therefore, we may write the short tensor notation $\mathbf{a} \cdot \mathbf{b} = a_i b_i$, meaning expanded and decoded $\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i$.

# 3 The Permutation tensor

The *permutation tensor* (alias permutation symbol) is a three-index function sometimes also called the *alternating tensor* or the *signature*. There are several common notations for the symbol: some authors uses the Greek epsilon character $\epsilon_{ijk}$, others uses the curly variant $\varepsilon_{ijk}$ or the Latin lower case $e_{ijk}$. We will use the the latter, because we will follow GRINFELD and reserve the Greek epsilon notation for the LEVI-CIVITA symbol, cf. [1, p.134]. The permutation tensor is heavily used to define advanced tensorial objects such as the determinant, the curl or the cross product of vectors, cf. [17].

## 3.1 Definition

In two dimensions, the permutation tensor is defined by:

$$e_{ij} = \begin{cases} +1 & \text{if } (i,j) = (1,2) \\ -1 & \text{if } (i,j) = (2,1) \\ 0 & \text{if } i = j \end{cases}$$

In three dimensions, the permutation tensor is defined by:

$$e_{ijk} = \begin{cases} +1 & \text{if } (i,j,k) \text{ is } (1,2,3),(2,3,1), \text{ or } (3,1,2), \\ -1 & \text{if } (i,j,k) \text{ is } (3,2,1),(1,3,2), \text{ or } (2,1,3), \\ 0 & \text{if } i = j, \text{ or } j = k, \text{ or } k = i \end{cases}$$

In four dimensions (and also in 2D or 3D), the permutation tensor is defined by:

$$e_{ijkl} = \begin{cases} +1 & \text{if } (i,j,k,l) \text{ is an even permutation of } (1,2,3,4) \\ -1 & \text{if } (i,j,k,l) \text{ is an odd permutation of } (1,2,3,4) \\ 0 & \text{otherwise} \end{cases}$$

**Remark.** We interpret the permutation *symbol* as a tensor, so we call it the permutation *tensor*. Be warned: some authors call the permutation *tensor* the Levi-Civita symbol.

| LEXICON | *Math* | EIGENMATH |
|---|---|---|
| *permutation tensor*: | $e_{ijk}$ | `eijk` or `Eijk` |

## 3.2 Implementation

### 3.2.1 ... of the two dimensional permutation tensor `eij`

```
# EIGENMATH
-- permutation tensor als function E
Eij(i,j) = j-i
Eij(1,2)
Eij(1,1)
```

```
-- permutation tensor eij as table

eij = zero(2,2)                              -- container
for(i,1,2, for(j,1,2,  eij[i,j]=j-i ))       -- fill container

eij                                  -- the permutation tensor
eij[1,2]                             -- result on index pair (1,2)
```

EIGENMATH output:

$$e_{ij} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

$$1$$

- ▷ Click here to run the script.

**Exercise 3.** Define function `Eij(i,j)` using the math definition for $e_{ij}$ from above.

### 3.2.2 ... the three dimensional permutation tensor `eijk`

```
# EIGENMATH
-- permutation tensor as function, i.e.
-- eijk is 1 if (i,j,k) is an even permutation of (1,2,3),
--         -1 if (i,j,k) is an odd permutation of (1,2,3),
--          0 if any index is repeated.

Eijk(i,j,k) = test(
   or( (i,j,k)==(1,2,3), (i,j,k)==(2,3,1), (i,j,k)==(3,1,2)), +1,
   or( (i,j,k)==(3,2,1), (i,j,k)==(1,3,2), (i,j,k)==(2,1,3)), -1,
   or( i==j, j==k, k==i), 0 )

Eijk(1,2,3)
Eijk(2,1,3)
Eijk(2,2,1)
```

EIGENMATH output:

$$1$$

$$-1$$

$$0$$

Now the version with table access:

```
# EIGENMATH
eijk = zero(3,3,3)
n = 3
for(i,1,n,
```

```
  for(j,1,n,
   for(k,1,n, eijk[i,j,k]=(j-i)*(k-i)*(k-j)/2 ) ))

eijk

--test:
eijk[1,2,1]                    -- access to entry with index list (1,2,1)
a = eijk[1,2,3] + eijk[2,1,3]
a
b = eijk[1,2,3] * eijk[2,3,1]
b
```

EIGENMATH output:

$$e_{ijk} = \left[ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right]$$

$$0$$

$$a = 0$$

$$b = 1$$

- ▷ Click here to run the script.

**Remark.** The EIGENMATH output of tensor `eijk` is verified by the two visualizations of this tensor:

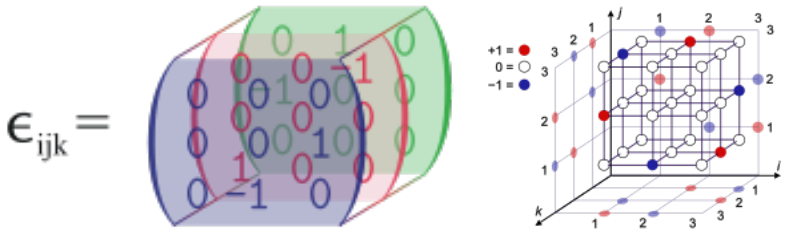



Figure 2: Left: 3D image of `eijk` with 3 layered matrix components. Right: values of `eijk` in a 3D coordinate cube in $\mathbb{R}^3$. Cf. both pictures borrowed from [17].

**Exercise 4.** Define function `Eijk(i,j,k)` using the EIGENMATH definition `eijk(i,j,k) = (j-i)*(k-i)*(k-j)/2`. Argue for the correctness of the formula and test it on some examples e.g. `eijk(3,2,1) = -1`.

**Remark.** The above formulas use the alternative explicit expression of the permutation tensor as a product of numbers using the *signum* function (denoted `sign`) as helper function:

$$\varepsilon_{a_1 a_2 a_3 \ldots a_n} = \prod_{1 \le i < j \le n} \operatorname{sign}(a_j - a_i)$$

**Exercise 5.** G. Weigt, the author of EIGENMATH, defined `eijk` explicit in this way:

```
# EIGENMATH (G. Weigt)
eijk = zero(3,3,3)
  eijk[1,2,3] =  1
  eijk[2,3,1] =  1
  eijk[3,1,2] =  1
  eijk[3,2,1] = -1
  eijk[1,3,2] = -1
  eijk[2,1,3] = -1

eijk   -- the tensor
```

Explain.

### 3.2.3 ... the 4D permutation tensor `eijkl`

In what follows we will not use more than the 4 dimensional version of the permutation tensor.

```
# EIGENMATH
-- permutation tensor eijkl as TABLE
eijkl=zero(4,4,4,4)
n=4
for(i,1,n, for(j,1,n, for(k,1,n, for(l,1,n,
        eijkl[i,j,k,l] = (i-j)*(i-k)*(i-l)*(j-k)*(j-l)*(k-l)/12 ) )))

eijkl
eijkl[3,4,2,1]   -- -1
eijkl[1,2,3,4]   --  1
eijkl[1,2,1,4]   --  0   ok

-- permutation tensor eijkl as FUNCTION
Eijkl(i,j,k,l) = (i-j)*(i-k)*(i-l)*(j-k)*(j-l)*(k-l)/12

Eijkl(3,4,2,1)   -- -1
Eijkl(1,2,3,4)   --  1
Eijkl(1,2,1,4)   --  0   ok
```

EIGENMATH output:

$$e_{ijkl} = \left[\begin{bmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{bmatrix} \begin{bmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&1\\0&0&-1&0 \end{bmatrix} \begin{bmatrix} 0&0&0&0\\0&0&0&-1\\0&0&0&0\\0&1&0&0 \end{bmatrix} \begin{bmatrix} 0&0&0&0\\0&0&1&0\\0&-1&0&0\\0&0&0&0 \end{bmatrix}\right.$$
$$\begin{bmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&-1\\0&0&1&0 \end{bmatrix} \begin{bmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{bmatrix} \begin{bmatrix} 0&0&0&1\\0&0&0&0\\0&0&0&0\\-1&0&0&0 \end{bmatrix} \begin{bmatrix} 0&0&-1&0\\0&0&0&0\\1&0&0&0\\0&0&0&0 \end{bmatrix}$$
$$\begin{bmatrix} 0&0&0&0\\0&0&0&1\\0&0&0&0\\0&-1&0&0 \end{bmatrix} \begin{bmatrix} 0&0&0&-1\\0&0&0&0\\0&0&0&0\\1&0&0&0 \end{bmatrix} \begin{bmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{bmatrix} \begin{bmatrix} 0&1&0&0\\-1&0&0&0\\0&0&0&0\\0&0&0&0 \end{bmatrix}$$
$$\left.\begin{bmatrix} 0&0&0&0\\0&0&-1&0\\0&1&0&0\\0&0&0&0 \end{bmatrix} \begin{bmatrix} 0&0&1&0\\0&0&0&0\\-1&0&0&0\\0&0&0&0 \end{bmatrix} \begin{bmatrix} 0&-1&0&0\\1&0&0&0\\0&0&0&0\\0&0&0&0 \end{bmatrix} \begin{bmatrix} 0&0&0&0\\0&0&0&0\\0&0&0&0\\0&0&0&0 \end{bmatrix}\right]$$

- ▷ Click here to open this file.

♡ The last output should convince you of the magnificent cool suitability of EIGENMATH for doing tensor calculus without pain - and to present tensors visually to inspect their action.

**Exercise 6.** Define the permutation tensor `eijkl` in EIGENMATH using the G. WEIGT-way, i.e. use a table of just enough prescribed values at quadruple index positions - and to make tensors printed visually to inspect their action.
[Hint: look carefully into the last EIGENMATH output. How many entrys do you need?]

## 3.3 Applications

As a proof of concept we show how the mathematical concepts of the determinant of a square matrix, the cross product of two vectors and the curl of a vector field can be defined and calculated using tensor technics with EIGENMATH.

### 3.3.1 Determinants

If $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ is a n-by-n square matrix, the determinant is defined and calculated through the so-called LEIBNIZ formula using sigma notation,

$$\det(A) = \sum_{\sigma\in S_n} \operatorname{sgn}(\sigma) a_{1,\sigma_1} \cdots a_{n,\sigma_n}$$

Using pi notation, this is in short $\det A = \sum_{\sigma\in S_n} \left(\operatorname{sgn}(\sigma) \prod_{i=1}^{n} a_{i,\sigma(i)}\right)$. With the permutation tensor, the Leibniz formula may be written as $\det(A) = \sum_{i_1,i_2,\ldots,i_n} \varepsilon_{i_1\cdots i_n} a_{1,i_1} \cdots a_{n,i_n}$, where the sum is taken over all n-tuples of integers in $\{1,\ldots,n\}$.

This is a horrible formula, which no one really wants to compute this way by paper'n pencil. So let's do EIGENMATH the work for us.

For the special case of a $3 \times 3$ matrix $A$ we have

$$\det(\mathbf{A}) = \sum_{i=1}^{3}\sum_{j=1}^{3}\sum_{k=1}^{3} \varepsilon_{ijk} a_{1i} a_{2j} a_{3k}$$

Here the permutation tensor $\varepsilon_{ijk}$ acts as a filter, which sort out the proper ones of the whole 27 summands.

```
# EIGENMATH

A=((1,2,3),(4,5,6),(7,8,8))        -- a 3-by-3 matrix

Eijk(i,j,k) = (j-i)*(k-i)*(k-j)/2  -- permutation tensor

Det(A) = sum(i,1,3,
         sum(j,1,3,
         sum(k,1,3,  Eijk(i,j,k)*A[1,i]*A[2,j]*A[3,k] ) ))

Det(A)       -- determinant via LEIBNIZ formula
det(A)       -- build-in det-function
```

EIGENMATH output: 3

- ▷ Click here to run this script.

**Exercise 7.** Define an EIGENMATH function `Det2` analog to the example above and calculate the determinant of matrix $A = ((1,2),(7,8))$.

### 3.3.2 Cross product

Given two linearly independent vectors $a$ and $b$ in $\mathbb{R}^3$, the cross product $a \times b$ is a vector, that is perpendicular to both $a$ and $b$, i.e. therefore perpendicular to the plane containing them. Using column vector notation, we have the explicit defining formula for $c = a \times b$:

$$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix}$$

In any basis, the cross-product $a \times b$ is given by the tensorial formula

$$(a \times b)_k = E_{ijk} a^i b^j$$

EINSTEIN ... ↓ ... expanded

$$(a \times b)_k = \sum_{i=1}^{3}\sum_{j=1}^{3} E_{ijk} a^i b^j$$

i.e. with the suppressed summation expanded, we get the correct formula for the translation of the cross product term into EIGENMATH code. Therein $E_{ijk}$ is the permutation tensor, which respects the positions of the indices. So we have:

```
# EIGENMATH
Eijk(i,j,k) = (j-i)*(k-i)*(k-j)/2          -- permutation tensor

do( E1=(1,0,0), E2=(0,1,0), E3=(0,0,1))    -- basis in R^3
E=(E1,E2,E3)

Cross(a,b) = sum(i,1,3,
               sum(j,1,3, sum(k,1,3,       -- implicit EINSTEIN summation
                 Eijk(i,j,k) * E[i]*a[j]*b[k] ))) -- the tensorial formula

Cross((1,2,3),(4,5,6))      -- test tensorial Cross product formula
cross((1,2,3),(4,5,6))      -- check using build-in cross
```

EIGENMATH output: `[-3, 6, -3]`

- ▷ Click here to run this script.

**Exercise 8.** If you only need the first component `axb1` of the cross product of $a$ and $b$, you can do: `axb1 = sum(j,1,3, sum(k,1,3, Eijk(1,j,k)*a[j]*b[k]))`. Verify it.

**Exercise 9.** Program the cross product using the explicit formula given above.

### 3.3.3 Curl

*curl* $F$ [2] is a vector operator that is used in vector calculus to describe the circulation of a 3D differentiable vector field $F\colon \mathbb{R}^3 \to \mathbb{R}^3$, given through its three component functions $F(x,y,z) = (F_x(x,y,z), F_y(x,y,z), F_z(x,y,z))$. We then have the definition:

$$(curl\, F)(x,y,z) = \left( \frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z}, \frac{\partial F_x}{\partial z} - \frac{\partial F_z}{\partial x}, \frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y} \right)$$

The equation for each component $(curl\, F)_k$ is obtained by exchanging each occurrence of a subscript $y, z, x$ in cyclic permutation $(y,z) \to (z,x) \to (x,y)$ by focussing at the denominators.

*Implementation* **N°1:** (classic vectorial version) We implement the classic definition from above in EIGENMATH and test it with the vector field $F(x,y,z) = (xy, -\sin(z), 1)$:

[2] In classic scientific literature in Europe, the alternative notation $rotF$ (read: rotation F) is used.

```
# EIGENMATH
Curl(F) = do(
     Cx = d(F[3],y)-d(F[2],z),
     Cy = d(F[1],z)-d(F[3],x),
     Cz = d(F[2],x)-d(F[1],y),
     C  = (Cx,Cy,Cz),
     C)

F = (x*y,-sin(z),1)    -- example vector field [Marsden, p.917]

Curl(F)     -- testing the definition
curl(F)     -- check against build-in curl
```

EIGENMATH output: `[cos(x), 0, -x]`

• Mark & Copy the blue code lines. ▷ Then Click here and Paste it into the input box ☐RUN. Press RUN.

*Implementation* **N$^o$2:** (tensorial version) If we write $F = (F_1, F_2, F_3)$ as a function of position $x = (x^1, x^2, x^3)$ in $\mathbb{R}^3$ using index notion, then the $i$th component of the curl of $F$ equals (summation suppressed according the EINSTEIN summation convention!)

$$(curl\,\mathbf{F})_i(\mathbf{x}) = \varepsilon_{ijk}\frac{\partial}{\partial x^j}F^k(\mathbf{x})$$

in tensorial notation, which follows from the cross product definition in 3.2.2, substituting components of the gradient vector. Here we do not need any memorizing help.

```
# EIGENMATH
F = (x*y, -sin(z), 1)   -- differential vector field in R^3
X = (x,y,z)             -- coordinates in R^3

Eijk(i,j,k) = (j-i)*(k-i)*(k-j)/2  -- permutation tensor

Curl(F,i) = sum(j,1,3,  -- ith component of curl
            sum(k,1,3,  Eijk(i,j,k) * d(F[k],X[j]) ))

Curl(F,1)               -- 1st component of curl F
Curl(F,2)
Curl(F,3)

CurlF = (Curl(F,1), Curl(F,2), Curl(F,3))
CurlF                   -- curl F as vector field
```

EIGENMATH output: `[cos(x), 0, -x]`

• ▷ Click here to run this script.

*Implementation* **N$^o$3:** (using tensor operators) The following wonderful compact implementation is by G. WEIGT. He use again the $i$th component tensor formula of the curl of $F$, i.e. $(curl\,\mathbf{F})_i(\mathbf{x}) = \varepsilon_{ijk}\frac{\partial}{\partial x^j}F^k(\mathbf{x})$. Because we call the tensor operators `contract(.)` and `outer(.)`, we have to use the (multidimensional) matrix table version `eijk` of the permutation tensor. We explain the following two constructions `curl1` and `curl2` later, but would like to whet your appetite for the following chapter, where these tensor operators incl. the `Kronecker` product $\otimes$ are themed.

```
# EIGENMATH
eijk = zero(3,3,3)      -- Define permutation tensor as TABLE
       eijk[1,2,3] =  1
       eijk[2,3,1] =  1
       eijk[3,1,2] =  1
       eijk[3,2,1] = -1
       eijk[1,3,2] = -1
       eijk[2,1,3] = -1

F = (x*y, -sin(z), 1)   -- differential vector field in R^3

curl1(F) = contract(contract(outer(eijk, d(F,(x,y,z))), 3,4),2,3)
curl1(F)

curl2(F) = contract(dot(eijk, d(F,(x,y,z))), 2,3)
curl2(F)
```

EIGENMATH output: `[cos(x), 0, -x]`

- ▷ Click here to run this script.

**Exercise 10.** Calculate the the curl of the following vector fields [Mardsen, p. 914 ff].
a. $F(x,y,z) = (e^z, -\cos(xy), z^3y)$.
b. $F(x,y,z) = (xy\cos(x), -yz\sin(x), -xy\tan(y))$.
c. $F(x,y,z) = (yz, -xz, xy)\cdot 1/(x^2+y^2+z^2)$.
d. $F(x,y,z) = (ye^z, xe^z, xye^z)$.

**Exercise 11.** Let $f\colon \mathbb{R}^3 \to \mathbb{R}$ be defined through $f(x,y,z) = x^2yz^3$ and $A\colon \mathbb{R}^3 \to \mathbb{R}^3$ be defined through $A(x,y,z) = (xz, -y^2, 2x^2y)$.
Calculate curl$(A)$ and curl$(f\cdot A)$, cf. [8, p. 151].

**Exercise 12.** Let $f$ be $f(x,y,z) = xy + yz + zx$ and $A(x,y,z) = (x^2y, y^2z, z2^{)}$.
Calculate curl$(A)$ and curl$(f\cdot A)$ at the point $P = (3,-1,2)$, cf. [8, p. 151].

**Exercise 13.** If $A\colon \mathbb{R}^3 \to \mathbb{R}^3$ is defined through $A(x,y,z) = (3xz^2, -yz, x+2z)$, calculate curl curl$(A)$, cf. [8, p. 158].

# 4 The KRONECKER product and the outer product

## 4.1 outer

EIGENMATH's outer function implements the tensor operator $\overset{outer}{\otimes}$ of tensor algebra. Given two tensors (i.e. multidimensional 'arrays' or tables of numbers), their outer product is again a tensor. The outer product of the tensors $T$ and $S$ is a tensor with dimension $(dim(T), dim(S))$. We define the outer product in the special case of vectors, leave the definition of the outer product of matrices for the reader, but give the definition for the general case of two tensors, which includes the case of matrices, cf. [19].

### 4.1.1 Definition of outer

**a.** Given two vectors $\mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_m \end{bmatrix}$ and $\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}$ of size $m \times 1$ and $n \times 1$, their *outer product*, denoted $\mathbf{u} \overset{outer}{\otimes} \mathbf{v}$ is defined as the $m \times n$ matrix obtained by multiplying each element of $\mathbf{u}$ by each element of $\mathbf{v}$ :

$$\mathbf{u} \overset{outer}{\otimes} \mathbf{v} = \begin{bmatrix} u_1 v_1 & u_1 v_2 & \dots & u_1 v_n \\ u_2 v_1 & u_2 v_2 & \dots & u_2 v_n \\ \vdots & \vdots & \ddots & \vdots \\ u_m v_1 & u_m v_2 & \dots & u_m v_n \end{bmatrix}$$

In tensorial index notation: $(\mathbf{u} \overset{outer}{\otimes} \mathbf{v})_{ij} = u_i v_j$

**b.** Given two tensors $\mathbf{T}$ and $\mathbf{S}$ with dimensions $(k_1, k_2, \dots, k_m)$ and $(l_1, l_2, \dots, l_n)$, their outer product $\mathbf{T} \overset{outer}{\otimes} \mathbf{S}$ is a tensor with dimensions $(k_1, k_2, \dots, k_m, l_1, l_2, \dots, l_n)$ and entries $(\mathbf{T} \otimes \mathbf{S})_{i_1,i_2,\dots i_m,j_1,j_2,\dots,j_n} = u_{i_1,i_2,\dots,i_m} \cdot v_{j_1,j_2,\dots,j_n}$

The concept of the outer product $\overset{outer}{\otimes}$ is best understood by looking at several examples.

### 4.1.2 Examples of the use of outer

**Example 2.** (the outer product of two vectors)

```
# EIGENMATH
u=(1,2,3)
v=(3,2,1)
outer(u,v)
```

• Mark & Copy the blue code lines. ▷ Then Click here and Paste it into the input box ☐RUN. Press RUN.

EIGENMATH output:

$$\begin{bmatrix} 3 & 2 & 1 \\ 6 & 4 & 2 \\ 9 & 6 & 3 \end{bmatrix}$$

**Example 3.** (the outer product of two matrices with *arbitrary* dimensions)

```
# EIGENMATH
A = ((1, 2, 3),( 4, 5, 6))   -- a 2-by-3 matrix
A
B = ((1,1),(1,1))            -- ones(2), a 2-by-2 matrix
B
outer(A,B)                   -- a (2,3,2,2) tensor (multidim. matrix)
```

• Mark & Copy the blue code lines. ▷ Then Click here and Paste it into the input box ☐RUN. Press RUN.
EIGENMATH output:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

$$\begin{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} & \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix} & \begin{bmatrix} 3 & 3 \\ 3 & 3 \end{bmatrix} \\ \begin{bmatrix} 4 & 4 \\ 4 & 4 \end{bmatrix} & \begin{bmatrix} 5 & 5 \\ 5 & 5 \end{bmatrix} & \begin{bmatrix} 6 & 6 \\ 6 & 6 \end{bmatrix} \end{bmatrix}$$

Obviously `outer` respects the block structure of both input matrices.

**Example 4.** (the outer product of two tensors with *arbitrary* dimensions)

```
# EIGENMATH
i = quote(i)          -- clear name i (otherwise i^2=1 ;)

T = zero(2,2,2)       -- a (2,2,2) tensor
for(i,1,2,   for(j,1,2,  for(k,1,2,  T[i,j,k] = i^2*j*k)))   -- initialize T

S = zero(2,2,2,2)     -- a (2,2,2,2) tensor
for(i,1,2,  for(j,1,2,  for(k,1,2,  for(l,1,2, S[i,j,k,l] = i^2*j*k*l))))

T
S
TT = outer(T,T)
TT
TS=outer(T,S)
TS
```

• ▷ Click here to run this script.
EIGENMATH output:

$$T = \begin{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \\ \begin{bmatrix} 4 & 8 \\ 8 & 16 \end{bmatrix} \end{bmatrix}$$

$$S = \begin{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} & \begin{bmatrix} 2 & 4 \\ 4 & 8 \end{bmatrix} \\ \begin{bmatrix} 4 & 8 \\ 8 & 16 \end{bmatrix} & \begin{bmatrix} 8 & 16 \\ 16 & 32 \end{bmatrix} \end{bmatrix}$$

$$T_T = \begin{bmatrix} \begin{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} & \begin{bmatrix} 4 & 8 \\ 8 & 16 \end{bmatrix} \\ \begin{bmatrix} 2 & 4 \\ 4 & 8 \end{bmatrix} & \begin{bmatrix} 8 & 16 \\ 16 & 32 \end{bmatrix} \end{bmatrix} & \begin{bmatrix} \begin{bmatrix} 2 & 4 \\ 4 & 8 \end{bmatrix} & \begin{bmatrix} 8 & 16 \\ 16 & 32 \end{bmatrix} \\ \begin{bmatrix} 4 & 8 \\ 8 & 16 \end{bmatrix} & \begin{bmatrix} 16 & 32 \\ 32 & 64 \end{bmatrix} \end{bmatrix} \\ \begin{bmatrix} \begin{bmatrix} 4 & 8 \\ 8 & 16 \end{bmatrix} & \begin{bmatrix} 16 & 32 \\ 32 & 64 \end{bmatrix} \\ \begin{bmatrix} 8 & 16 \\ 16 & 32 \end{bmatrix} & \begin{bmatrix} 32 & 64 \\ 64 & 128 \end{bmatrix} \end{bmatrix} & \begin{bmatrix} \begin{bmatrix} 8 & 16 \\ 16 & 32 \end{bmatrix} & \begin{bmatrix} 32 & 64 \\ 64 & 128 \end{bmatrix} \\ \begin{bmatrix} 16 & 32 \\ 32 & 64 \end{bmatrix} & \begin{bmatrix} 64 & 128 \\ 128 & 256 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Here is tensor $TS$:

$$T_S = \begin{bmatrix} \begin{bmatrix} \begin{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} & \begin{bmatrix} 2 & 4 \\ 4 & 8 \end{bmatrix} \\ \begin{bmatrix} 4 & 8 \\ 8 & 16 \end{bmatrix} & \begin{bmatrix} 8 & 16 \\ 16 & 32 \end{bmatrix} \end{bmatrix} & \begin{bmatrix} \begin{bmatrix} 2 & 4 \\ 4 & 8 \end{bmatrix} & \begin{bmatrix} 4 & 8 \\ 8 & 16 \end{bmatrix} \\ \begin{bmatrix} 8 & 16 \\ 16 & 32 \end{bmatrix} & \begin{bmatrix} 16 & 32 \\ 32 & 64 \end{bmatrix} \end{bmatrix} \\ \begin{bmatrix} \begin{bmatrix} 2 & 4 \\ 4 & 8 \end{bmatrix} & \begin{bmatrix} 4 & 8 \\ 8 & 16 \end{bmatrix} \\ \begin{bmatrix} 8 & 16 \\ 16 & 32 \end{bmatrix} & \begin{bmatrix} 16 & 32 \\ 32 & 64 \end{bmatrix} \end{bmatrix} & \begin{bmatrix} \begin{bmatrix} 4 & 8 \\ 8 & 16 \end{bmatrix} & \begin{bmatrix} 8 & 16 \\ 16 & 32 \end{bmatrix} \\ \begin{bmatrix} 16 & 32 \\ 32 & 64 \end{bmatrix} & \begin{bmatrix} 32 & 64 \\ 64 & 128 \end{bmatrix} \end{bmatrix} \end{bmatrix} \\ \begin{bmatrix} \begin{bmatrix} \begin{bmatrix} 4 & 8 \\ 8 & 16 \end{bmatrix} & \begin{bmatrix} 8 & 16 \\ 16 & 32 \end{bmatrix} \\ \begin{bmatrix} 16 & 32 \\ 32 & 64 \end{bmatrix} & \begin{bmatrix} 32 & 64 \\ 64 & 128 \end{bmatrix} \end{bmatrix} & \begin{bmatrix} \begin{bmatrix} 8 & 16 \\ 16 & 32 \end{bmatrix} & \begin{bmatrix} 16 & 32 \\ 32 & 64 \end{bmatrix} \\ \begin{bmatrix} 32 & 64 \\ 64 & 128 \end{bmatrix} & \begin{bmatrix} 64 & 128 \\ 128 & 256 \end{bmatrix} \end{bmatrix} \\ \begin{bmatrix} \begin{bmatrix} 8 & 16 \\ 16 & 32 \end{bmatrix} & \begin{bmatrix} 16 & 32 \\ 32 & 64 \end{bmatrix} \\ \begin{bmatrix} 32 & 64 \\ 64 & 128 \end{bmatrix} & \begin{bmatrix} 64 & 128 \\ 128 & 256 \end{bmatrix} \end{bmatrix} & \begin{bmatrix} \begin{bmatrix} 16 & 32 \\ 32 & 64 \end{bmatrix} & \begin{bmatrix} 32 & 64 \\ 64 & 128 \end{bmatrix} \\ \begin{bmatrix} 64 & 128 \\ 128 & 256 \end{bmatrix} & \begin{bmatrix} 128 & 256 \\ 256 & 512 \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

**Exercise 14.** Let $A = ((x, y), (z, u))$ and $B = ((a, b), (c, d))$.
Calculate $A \overset{outer}{\otimes} B$, $B \overset{outer}{\otimes} (A \overset{outer}{\otimes} B)$ and $(A \overset{outer}{\otimes} B) \overset{outer}{\otimes} (A \overset{outer}{\otimes} B)$
using the build-in function `outer`. Imagine by mind what output is to be expected.

**Exercise 15.** (The inner product is the contract of the outer product.)
Given two vectors $\mathbf{u} = (u_1, u_2, \ldots, u_m)$ and $\mathbf{v} = (v_1, v_2, \ldots, v_n)$.
Verify: The dot product $\mathbf{u} \bullet \mathbf{v}$=`dot(u,v)` is the contract (trace) of the outer product.

**Exercise 16.** Think about the algebraic properties of the outer product $\overset{outer}{\otimes}$.
Chheck, if it is associative, commutative, bilinear etc.

## 4.2 kronecker

EIGENMATH's `kronecker` function implements the tensor operator $\otimes$ of tensor algebra, i.e. `kronecker(A,B)` returns the Kronecker tensor product of tensors (e.g. matrices) $A$ and $B$. The Kronecker product is best understood by thinking about block matrices. If $A$ is a $n \times p$ matrix and $B$ is a $m \times q$ matrix, then the Kronecker product $A \otimes B$ is a block matrix that is build from blocks of $B$, where each block is multiplied by an element of $A$, cf. [16].

### 4.2.1 Definition of kronecker

Let's begin with the inevitable theoretical mathematical definition.
Let $A$ be an $m \times n$ matrix and let $B$ an $p \times q$ matrix. Then the *Kronecker product of $A$ and $B$*, symbolically written $A \otimes B$, is the $m \cdot p \times n \cdot q$ matrix explicitly defined by

$$A \otimes B = (a_{ij} \cdot B) = \begin{pmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{pmatrix}$$

More descriptive: If $A$ is an m-by-n matrix and $B$ is a p-by-q matrix, then `kronecker(A,B)` is an $m \cdot p$-by-$n \cdot q$ matrix, which is calculated by taking all possible products between the elements of $A$ and those of $B$, i.e. the Kronecker tensor product of $A$ and $B$ is a large matrix formed by multiplying $B$ by each element of $A$.

**Example 5.**

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \otimes \begin{pmatrix} 7 & 8 \\ 9 & 0 \end{pmatrix} = \begin{pmatrix} 1 \cdot \begin{pmatrix} 7 & 8 \\ 9 & 0 \end{pmatrix} & 2 \cdot \begin{pmatrix} 7 & 8 \\ 9 & 0 \end{pmatrix} \\ 3 \cdot \begin{pmatrix} 7 & 8 \\ 9 & 0 \end{pmatrix} & 4 \cdot \begin{pmatrix} 7 & 8 \\ 9 & 0 \end{pmatrix} \\ 5 \cdot \begin{pmatrix} 7 & 8 \\ 9 & 0 \end{pmatrix} & 6 \cdot \begin{pmatrix} 7 & 8 \\ 9 & 0 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 7 & 8 & 14 & 16 \\ 9 & 0 & 18 & 0 \\ 21 & 24 & 28 & 32 \\ 27 & 0 & 36 & 0 \\ 35 & 40 & 42 & 48 \\ 45 & 0 & 54 & 0 \end{pmatrix}$$

### 4.2.2 Examples of the use of kronecker

```
# EIGENMATH input:
kronecker( ((1,2),(3,4),(5,6)), ((7,8),(9,0)) )
```

EIGENMATH output:

cf. example 5.

We now repeat the examples of `outer` to contrast them with the results of `kronecker`.

**Example 6.** (the Kronecker product of two vectors)

```
# EIGENMATH
u=(1,2,3)
v=(3,2,1)
kronecker(u,v)
```

• Mark & Copy the blue code lines. ▷ Then Click here and Paste it into the input box □RUN. Press RUN.
EIGENMATH output:

$$\begin{bmatrix} 3 \\ 2 \\ 1 \\ 6 \\ 4 \\ 2 \\ 9 \\ 6 \\ 3 \end{bmatrix}$$

In contrast to `outer` the output of `kronecker` has stacked the rows to one long column.

**Example 7.** (the Kronecker product of two matrices with *arbitrary* dimensions)

```
# EIGENMATH
A = ((1, 2, 3),( 4, 5, 6))   -- a 2-by-3 matrix
A
B = ((1,1),(1,1))            -- ones(2), a 2-by-2 matrix
B
kronecker(A,B)                 -- a (2,3,2,2) tensor (multidim. matrix)
```

• Mark & Copy the blue code lines. ▷ Then Click here and Paste it into the input box □RUN. Press RUN.

EIGENMATH output:

$$\begin{bmatrix} 1 & 1 & 2 & 2 & 3 & 3 \\ 1 & 1 & 2 & 2 & 3 & 3 \\ 4 & 4 & 5 & 5 & 6 & 6 \\ 4 & 4 & 5 & 5 & 6 & 6 \end{bmatrix}$$

In contrast to `outer` the output of `kronecker` has cleared the brackets of the inner matrices to give one single matrix. Obviously `kronecker` 'flattens' the block structure of both input matrices.

**Remark.** The KRONECKER product of two tensors with *arbitrary* dimensions is currently not implemented in EIGENMATH. So there is no return value e.g for the tensor product $(T[i,j,k] = i^2 * j * k) \otimes (S[i,j,k,l] = i^2 * j * k * l)$.

**Exercise 17.** Let $A = ((x, y), (z, u))$ and $B = ((a, b), (c, d))$.
Calculate $A \otimes B$, $B \otimes (A \otimes B)$ and $(A \otimes B) \otimes (A \otimes B)$

**Exercise 18.** Write a function `ones(n)`, which returns a n-by-n matrix whose all entries are 1. E.g.

```
# EIGENMATH
n =2
ones(n)
```

EIGENMATH output: ((1,1),(1,1))

## 4.3 Applications of `kronecker`

### 4.3.1 The direct sum $\oplus$

Let $A$ be an $m \times n$ matrix and $B$ an $p \times q$ matrix. Then the *direct sum of $A$ and $B$*, symbolically written $A \oplus B$, is the $m \cdot p \times n \cdot q$ matrix explicitly defined by

$$A \oplus B = \begin{pmatrix} A & O \\ O & B \end{pmatrix}$$

i.e. for any arbitrary matrices $A$ and $B$ the direct sum (aka tensor sum) is defined as the block diagonal matrix of $A$ and $B$. Both zero matrices $O$ are of appropriate dimension.

**Example 8.** (cf. [9, p. 7], [11, p. 28])

$$\begin{pmatrix} 1 & 2 & 3 \end{pmatrix} \oplus \begin{pmatrix} 4 & 5 \\ 6 & 7 \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} 1 & 2 & 3 \end{pmatrix} & \begin{pmatrix} 0 & 0 \end{pmatrix} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} & \begin{pmatrix} 4 & 5 \\ 6 & 7 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 & 0 & 0 \\ 0 & 0 & 0 & 4 & 5 \\ 0 & 0 & 0 & 6 & 7 \end{pmatrix}$$

We show how the direct sum could be used in EIGENMATH using function `kronecker`.

```
# EIGENMATH

A=((1,2),(3,4))
B=((4,5),(6,7))
U=((1,0),   -- Upper
   (0,0))
L=((0,0),   -- Lower
   (0,1))

kronecker(U,A)

directSum(A,B)= kronecker(U,A) + kronecker(L,B)
directSum(A,B)
```

• ▷ Click here to run this script.
EIGENMATH output:

$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 2 & 0 & 0 \\ 3 & 4 & 0 & 0 \\ 0 & 0 & 4 & 5 \\ 0 & 0 & 6 & 7 \end{bmatrix}$$

The first matrix is the result of the call `kronecker(U,A)` and the second matrix is the result of the call `directSum(A,B)`

**Exercise 19.** Calculate the direct sum of $A$ and $B$ from example 9 using `directSum(A,B)`

### 4.3.2 Construction of the basis of $\mathbb{R}^4$ from the basis of $\mathbb{R}^2$, cf. [9, p. 7]

Let $u = (1, 0), v = (0, 1)$ be the canonical standard basis in $\mathbb{R}^2$. Verify, that $(u \otimes u, u \otimes v, v \otimes u, u \otimes u)$ is the standard basis in $\mathbb{R}^2$.
We use EIGENMATH's `kronecker(.)`.

```
# EIGENMATH
kron(A,B)=kronecker(A,B)

u=(1,0)
v=(0,1)

-- Basis R^4
kron(u,u)
kron(u,v)
kron(v,u)
kron(v,v)
```

• Mark & Copy the blue code lines. ▷ Then Click here and Paste it into the input box ☐RUN. Press RUN.
EIGENMATH output: $[1, 0, 0, 0], ..., [0, 0, 0, 1]$

**Exercise 20.** Construct an orthonormal basis of $\mathbb{C}^4$ with the help of `kronecker`.
Hint: start with $u = 1/\sqrt{2} \cdot (1, i)$ and $u = 1/\sqrt{2} \cdot (1, -i)$.

**Exercise 21.** (cf. [12, p. 112]) A basis in the Hilbert space $M_2(\mathbb{C})$ of 2-by-2 matrices is given by the PAULI spin matrices

$$\sigma_0 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \sigma_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \sigma_2 = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \sigma_3 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

including the 2-by-2 identity matrix. Use this orthonormal basis and the Kronecker products $\sigma_j \otimes \sigma_k$ with $j, k = 0, 1, 2, 3$ to find a basis in the Hilbert space $M_4(\mathbb{C})$.

*Solution*:

```
# EIGENMATH
kron(A,B)=kronecker(A,B)
-- Pauli spins is basis in M2(C)
s(0)=((1,0),(0,1))
s(1)=((0,1),(1,0))
s(2)=((0,-i),(i,0))
s(3)=((1,0),(0,-1))

for(j,0,3,
    for(k,0,3,
        print( kron(s(j),s(k)) )   -- Basis M4(C)
))
```

- ▷ Click here to run this script.

**Exercise 22.** Do some more of the examples 2.1 . . . 2.12 in [12] using EIGENMATH.

# 5 contract

The *contraction* of a tensor acts by setting equal a pair of indices[3] of the tensor to each other and summed over. In Einstein notation this summation is implicit in the notation. The result is a new tensor with order reduced by 2, cf. [20].
Contraction of tensors is heavily used in Differential Geometry or in General Relativity. Tensor contraction is a generalization of the trace of a matrix.

### 5.0.1 Definition of `contract`

Let $\mathbf{T} = T^i{}_j$ be a tensor. Then its *contract*ion is

$$T^i{}_j\delta_i{}^j = T^j{}_j = \sum_{j=1}^{n} T^j{}_j = T^1{}_1 + \cdots + T^n{}_n$$

In EIGENMATH this is expressed with the call of `contract(T,i,j)`.

• A general contraction is denoted by labeling one (covariant, lower) index and one (contravariant, upper) index with the same letter, summation over that index being implied by the summation convention. The resulting contracted tensor inherits the remaining indices of the original tensor, cf. [20].
• For example, contracting a tensor T of type (2,2) on the second and third indices (i.e. $b$) to create a new tensor U of type (1,1) is written as

$$T^{ab}{}_{bc} = \sum_b T^{ab}{}_{bc} = T^{a1}{}_{1c} + T^{a2}{}_{2c} + \cdots + T^{an}{}_{nc} = U^a{}_c$$

**Example 9.**

```
# EIGENMATH
u=(1)
contract(u)

A = ((1, 2, 3),( 4, 5, 6))
-- contract(A)  -- unequal tensor dimensions !

B = ((1,2),(3,4))          -- contract = trace
contract(B)
```

EIGENMATH output:

$$1 \qquad error \qquad 5$$

**Comment.** `contract(B)` contracts ('adds') the diagonal elements of $B$, i.e. $1 + 4 = 5$. Tensor $B$ (a matrix of shape 2-by-2) is reduced into a new tensor of type 1-by-1, i.e. a number 5.

[3] one a subscript (upper) index, the other a superscript (lower) index

**Example 10.**

```
# EIGENMATH
i = quote(i)          -- clear i

T = zero(2, 2,2)
for(i,1,2,   for(j,1,2,  for(k,1,2,  T[i,j,k] = i^2*j*k)))
T
contract(T,1,2)        -- (1)
contract(T,2,3)        -- (2)
```

• ▷ Click here to run this script.

EIGENMATH output:

$$T = \begin{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \\ \begin{bmatrix} 4 & 8 \\ 8 & 16 \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} 9 \\ 18 \end{bmatrix}$$

$$\begin{bmatrix} 5 \\ 20 \end{bmatrix}$$

**Comment.** In (1) the command `contract(T,1,2)` adds $(1{+}8, 2{+}16) = (9, 18) = \left[\begin{smallmatrix}1\\2\end{smallmatrix}\right] + \left[\begin{smallmatrix}8\\16\end{smallmatrix}\right]$, i.e. it adds the 1st column of 1st submatrix to the 2nd column of the second submatrix. In (2) the command `contract(T,2,3)` adds $(1+4, 4+16) = (5, 20) = \left[\begin{smallmatrix}1\\4\end{smallmatrix}\right] + \left[\begin{smallmatrix}4\\16\end{smallmatrix}\right]$, i.e it adds the trace of 1st submatrix to the trace of the second submatrix. Tensor $T$ (an array of shape 2-by-2-by-2) is reduced into a new tensor of type 2-by-1, i.e. a vector.
Let's elaborate a bit on it. If we fill tensor $T$ with arbitrary values we are able to follow the path of picking the right entries of $T$ by `contract(.,1,2)`:

```
T = zero(2, 2,2)   -- an tensor with 2^3=8 general elements
"contract(T,1,2):"
-- tensor index
-- i j k
-- 1 2 3
-- . .        (..,1,2)  : choose all entries with 'i=j' aka '1=2'
--                       i.e. all entries of type   11x  and  22x.
T[1,1,1] = a  -- .
T[1,1,2] = b  -- ..
T[1,2,1] = c
T[1,2,2] = d
T[2,1,1] = e
T[2,1,2] = f
```

```
T[2,2,1] = g  -- .  = a + g
T[2,2,2] = h  -- .. = b + h
```

That is explicit: `contract(T,1,2)` $= \begin{bmatrix} \circ+\circ \\ \circ\circ+\circ\circ \end{bmatrix} = \begin{bmatrix} T[1,1,1]+T[2,2,1] \\ T[1,1,2]+T[2,2,2] \end{bmatrix}$.

Now look at `contract(.,2,3)`:

```
"contract(T,2,3):"
T = zero(2, 2,2)
-- tensor index
-- i j k
-- 1 2 3
-- . .        (..,2,3)  : choose all entries with 'j=k' aka '2=3'
--                        i.e. all entries of type    x11  and  x22.
T[1,1,1] = a  -- .
T[1,1,2] = b
T[1,2,1] = c
T[1,2,2] = d  -- .   = a + d
T[2,1,1] = e  -- ..
T[2,1,2] = f
T[2,2,1] = g
T[2,2,2] = h  -- .. = e + h
```

That is explicit: `contract(T,2,3)` $= \begin{bmatrix} \circ+\circ \\ \circ\circ+\circ\circ \end{bmatrix} = \begin{bmatrix} T[1,1,1]+T[1,2,2] \\ T[2,1,1]+T[2,2,2] \end{bmatrix}$.

**Example 11.**

```
# EIGENMATH
S = zero(2, 2,2,2)
for(i,1,2,  for(j,1,2,  for(k,1,2,  for(l,1,2,
    S[i,j,k,l] = i^2*j*k*l))))

S
contract(S)          -- (1), i.e. contract(S)=contract(S,1,2)
contract(S,2,3)      -- (2)
contract(S,1,3)      -- (3)
```

• ▷ Click here to run this script.

EIGENMATH output:

$$S = \begin{bmatrix} \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} & \begin{bmatrix} 2 & 4 \\ 4 & 8 \end{bmatrix} \\ \begin{bmatrix} 4 & 8 \\ 8 & 16 \end{bmatrix} & \begin{bmatrix} 8 & 16 \\ 16 & 32 \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} 9 & 18 \\ 18 & 36 \end{bmatrix}$$

$$\begin{bmatrix} 5 & 10 \\ 20 & 40 \end{bmatrix}$$

$$\begin{bmatrix} 9 & 18 \\ 18 & 36 \end{bmatrix}$$

**Comment.** In (1) and (3) `contract(S)`≡`contract(S,1,2)=contract(S,1,3)` adds $(1+8, 2+16) = (9, 18) = \begin{bmatrix}1\\2\end{bmatrix} + \begin{bmatrix}8\\16\end{bmatrix}$, i.e. it adds the 1st column of 1st 'sub'matrix to the 2nd column of the second 'sub'matrix.
In (2) the command `contract(S,2,3)` picks the trace of each matrix-part of $S$ and puts it as entry at the corresponding position in the new reduced tensor of type 2-by-2, i.e. a matrix.

**Exercise 23.** Watch the output of `contract` by looking at an arbitrary (2,2,2,2) tensor $S$ and elaborate a bit on it as in example 11. Therefore fill tensor $S$ with arbitrary values and follow the path of picking the right entries of $S$ by `contract(.,1,2)` and `contract(.,2,3)`:

```
# EIGENMATH
S = zero(2, 2,2,2)   -- we need 2^4=16 elements
S[1,1,1,1] =a
S[1,1,1,2] =b
-- ...
S[2,2,2,1] =o
S[2,2,2,2] =p
```

## 5.1 A Potpourri of Tensor Operators - the case of the *curl*

In chapter 3.2.3 we looked at the wonderful compact EIGENMATH implementation of the *curl* by G. WEIGT. We started with an arbitrary vector-valued function $F\colon \mathbb{R}^3 \to \mathbb{R}^3$ defined through its component functions $F = (F_1, F_2, F_3)$ in the coordinates $(x_1, x_2, x_3)$. Then we used the $i$th component tensorial formula of the curl of $F$, i.e.

$$(curl\,\mathbf{F})_i(\mathbf{x}) = \varepsilon_{ijk}\frac{\partial}{\partial x^j}F^k(\mathbf{x}) \tag{5.1}$$

*Task*: How to translate formula (5.1) to EIGENMATH?

*Solution*: Analyse formula (4.1). Index $i$ is 'free' in the defining term in the sense that it is not seen twice on the RHS of (4.1). But index $j$ and $k$ are to be seen twofold on the RHS - each one once as superscript ('upper index') and once as subscript ('lower index'). Therefore there are 2 implicit hidden summation processes involved in accordance with the EINSTEIN summation convention. So we have three phases $A \to B \to C$ to construct the term in EIGENMATH:

$$\varepsilon_{ijk} \cdot \frac{\partial}{\partial x^j}F^k(\mathbf{x})$$

$A \downarrow$ *outer to build products*

$B \downarrow$ *contract over k, i.e. sum*

$C \downarrow$ *contract over j, i.e. sum*

We are able to use EIGENMATH, because we know the ingredients, i.e. the tensor operators `contract`, `outer` and the (multidimensional) table version `eijk` of the permutation tensor.

```
# EIGENMATH
F = (F1(),F2(),F3())  -- the vector-valued generic function F=(F1,F2,F3)

eijk = zero(3,3,3)    -- the permutation tensor eijk
eijk[1,2,3] = 1
eijk[2,3,1] = 1
eijk[3,1,2] = 1
eijk[3,2,1] = -1
eijk[1,3,2] = -1
eijk[2,1,3] = -1

eijk

d(F,(x1,x2,x3))      -- the JACOBI matrix of F of type 3x3
```

$$\begin{pmatrix} d[F_1(),x_1] & d[F_1(),x_2] & d[F_1(),x_3] \\ d[F_2(),x_1] & d[F_2(),x_2] & d[F_2(),x_3] \\ d[F_3(),x_1] & d[F_3(),x_2] & d[F_3(),x_3] \end{pmatrix}$$

```
-- ------ A -> B -> C becomes the curl of F

A = outer(eijk, d(F,(x1,x2,x3)))
--            |
--       eijk , dF
--            |
--          outer
--            |
--      (eijk * dF_k /dx_j)
--        123      4     5

-- A is now a multi-matrix of 27 elements, i.e. a 3 x 3x3 Tensor
A      -- take a look at A

-- 1st contraction on k, which is at index positions 3 and 4
--             | |
B = contract(A,3,4)        -- i.e. EINSTEIN sum with index k
--      |
--      sum over k=3=4
--  Sum eij(k) dF_(k) /dx_j
--       12 .      .      3
--       j is now at index positions 2 and 3 (k is 'killed')

B      -- take a look at B

-- 2nd Contraction on j, which lives now at index positions 2 and 3
--             | |
C = contract(B,2,3)        -- i.e. EINSTEIN sum with indes j
--      |
--      sum over j=2=3
--  Sum eij(k) dF_(k) /dx_j
--       1. .      .      . (j is killed)
-- i.e. add (trace) all elements on the main diagonals of B
-- i.e. 0 + dF3/dx2 - dF2/dx3, ...

C                          -- is now the curl of F
```

- ▷ Click here to run this script.

$$C = \begin{bmatrix} -\mathrm{d}\left[F_2(),x_3\right] + \mathrm{d}\left[F_3(),x_2\right] \\ \mathrm{d}\left[F_1(),x_3\right] - \mathrm{d}\left[F_3(),x_1\right] \\ -\mathrm{d}\left[F_1(),x_2\right] + \mathrm{d}\left[F_2(),x_1\right] \end{bmatrix}$$

Shorted version:

```
--         ijk     k j            -- 0. index list by equation (4.1)
A = outer(eijk, d(F,(x1,x2,x3)))  -- 1. do all element-wise products
A       -- ..3     4 .
B = contract(A,3,4)               -- 2. sum across k
--         ij.     . j
B       -- 12.     . 3
C = contract(B,2,3)               -- 3. sum across j
C
```

Shortest version by concatenation of processes A-B-C:

```
curlF = contract(contract(outer(eijk, d(F,(x1,x2,x3))),3,4),2,3)
curlF
```

**Exercise 24.** Repeat the reasoning above using the concrete function $F = (ye^z, xe^z, xye^z)$.

# 6 The Metric Tensor

The so-called *metric tensor*[4] $g$ (or simply metric) plays an crucial role in Differential Geometry and General Relativity. It allows defining and calculate distances and angles on surfaces ('manifolds') in space, just as the inner product do it on a Euclidean space.
The components of a metric tensor $g$ in a coordinate basis is a symmetric matrix (tensor), whose entries transform 'covariantly' under changes to the coordinate system. Thus a metric tensor is a covariant symmetric tensor, cf.[18].

A metric tensor is heavily used to do so-called *index juggling*, cf. [1, p.88], i.e. the tensor operations of *raising* or *lowering* an index. In this context is is essential to keep track of the position of an index, i.e. if the index is *superscripted* ('upper index') or *subscripted* ('lower index') in a tensor expression. For this purpose, the terms *covariant* ('lower') resp. *contravariant* ('upper') index are introduced to distinguish between them and to use the EINSTEIN summation convention correctly.

The following lexicon helps to memorize thes conventions.

| LEXICON | *Math* | EIGENMATH |
|---|---|---|
| *covariant tensor T of type (0,2)*: | $T_{ij}$ | `Tdd` |
| *contravariant tensor T of type (2,0)*: | $T^{ij}$ | `Tuu` |
| *mixed-variant tensor T of type (1,1)*: | $T^i_j$ or $T^i_{.j}$ | `Tud` |

**Remark.** In EIGENMATH we try to mimic the tensor notion of ${}_{ij}$ resp. ${}^{ij}$ resp. ${}^{i}_{j}$ by the identifier `dd` (read: 'down down') resp. `uu` (read: 'up up') resp. `ud` (read: 'up down').[5]

## 6.1 Definition

Let $(\mathbf{Z_1}, .., \mathbf{Z_n})$[6] be an basis of the $n$-dimensional vector space $V$ with inner product $\bullet$.
Then, by definition, the (covariant) *metric tensor* $g_{ij}$ consists ot the pairwise dot products of the basis vectors, i.e.

$$g_{ij} := \mathbf{Z_i} \bullet \mathbf{Z_j}$$

**Remark.** The metric tensor is also called GRAM's matrix, therefore the notion $g_{ij}$.
In what follows remember this lexicon:

| LEXICON | *Math* | EIGENMATH |
|---|---|---|
| *metric tensor g*: | $g_{ij}$ | `gdd` |
| *inverse of metric tensor* $g^{-1}$: | $g^{ij}$ | `guu` |

[4] aka *first fundamental form* or *fundamental tensor*
[5] This cool convention was introduced by G. WEIGT in his script about the SCHWARZSCHILD metric tensor, cf. [15].
[6] We follow the notations in [1].

## 6.2 Implementation and Examples

**Example 12.** (The Metric $g$ in Cartesian coordinates in $\mathbb{R}^2$) The most elementary example is that of the two-dimensional Euclidean geometry, the Euclidean metric tensor, cf. [1, p.62]. Choose the canonical basis $E = (E_1, E_2) = (\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix})$, then we have $g = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$.

```
# EIGENMATH
-- basis in Euclidean vector space R^2
E=((1,0),(0,1))

-- metric tensor in R^2
gdd = zero(2,2)
for(i,1,2,
   for(j,1,2,
            gdd[i,j]= dot(E[i],E[j]) ))

gdd

-- Calculate guu, i.e. the inverse of gij

guu = inv(gdd)
guu
```

- ▷ Click here to run this script.

**Remark.** Remember: the tensor notion $g_{ij}$ resp. $g^{ij}$ is noted here as `gdd` (read: 'g down down') resp. `guu` (read: 'g up up').

EIGENMATH output:

$$g_{dd} = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$$

$$g_{uu} = \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$$

**Example 13.** (The metric tensor $g$ in affine coordinates in $\mathbb{R}^2$) The Euclidean metric in an affine coordinate systems with basis $Z_1 = (2, 0)$ and $Z_2 = (cos(\pi/3), sin(\pi/3))$ can be calculated as follows, cf. [1, p. 64].

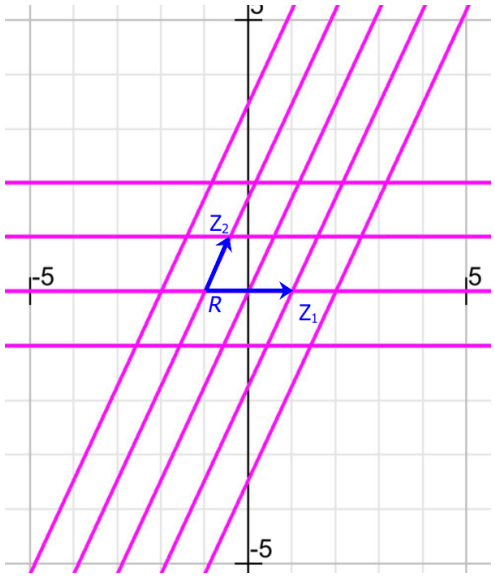



Figure 3: $R(x = -1, y = 0)$: position vector in ambient $\mathbb{R}^2$.
$Z_1$: 1st covariant basis vector at $R$.
$Z_2$: 2nd covariant basis vector at $R$.
///..: the coordinate grid corresponding basis $Z$.

```
# EIGENMATH
-- we introduce affine coordinates in R^2 by
Z1 = (2,0)
Z2 = (cos(pi/3),sin(pi/3))
Z1
Z2
-- the covariant basis in affine coordinates
Z=(Z1,Z2)

abs(Z1)
abs(Z2)

-- metric tensor in affine coordinates
gdd = zero(2,2)
for(i,1,2,
   for(j,1,2,
```

```
            gdd[i,j]= dot(Z[i],Z[j]) ))

gdd

-- Calculate guu, i.e. the inverse of gij

guu = inv(gdd)
guu

-- the contavariant basis is
E=((1,0),(0,1))
Zu1 = dot(guu, E[1])
Zu1

Zu2 = dot(guu, E[2])
Zu2
```

- ▷ Click here to run this script.

EIGENMATH output:

$$g_{dd} = \begin{bmatrix} 4 & 1 \\ 1 & 1 \end{bmatrix}$$

$$g_{uu} = \begin{bmatrix} \frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{4}{3} \end{bmatrix}$$

$$Z_{u1} = \begin{bmatrix} \frac{1}{3} \\ -\frac{1}{3} \end{bmatrix}$$

**Example 14.** (The metric tensor $g$ in polar coordinates in $\mathbb{R}^2$) The Euclidean metric in polar coordinates $(r, \theta)$ can be calculated as follows, cf. [1, p. 65]. Look at the position vector $R(r, \theta) = (r\cos(\theta), r\sin(\theta))$. Then the basis vectors $\mathbf{Z}_i$ are the tangential vectors on the two coordinate lines, i.e.

$$\mathbf{Z}_r = \frac{\partial R}{\partial r}$$
$$\mathbf{Z}_\theta = \frac{\partial R}{\partial \theta}$$

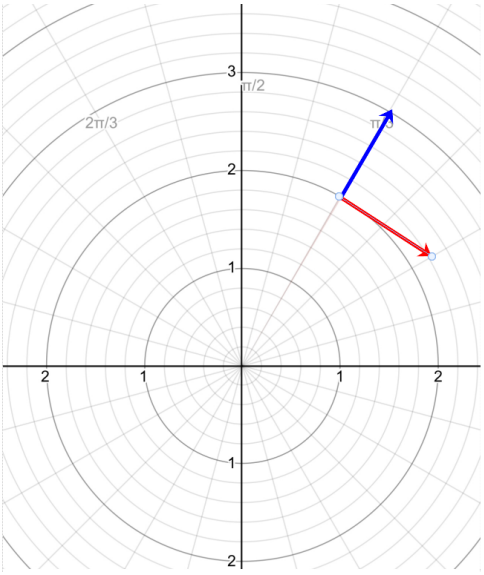



Figure 4: $R(r = 2, \theta = \pi/3)$: position vector in $\mathbb{R}^2$.
$\partial_r$: 1st covariant basis vector $Z_r$ at $R$.
$\partial_\theta$: 2nd covariant basis vector $Z_\theta$ at $R$.

Let's calculate the basis and the metric using

```
# EIGENMATH
-- we introduce polar coordinates in R^2 by
R(r,theta) = (r cos(theta),r sin(theta))
R

-- the covariant basis in polar coordinates
Z = d(R,(r,theta))    -- both basis vectors are in the columns
Z = transpose(Z)      -- therefore we have to transpose
Z

-- metric tensor gij in polar coordinates
```

```
gdd = zero(2,2)
for(i,1,2,
   for(j,1,2,
            gdd[i,j]= dot(Z[i],Z[j]) ))

gdd
gdd = simplify(gdd)
gdd

-- Calculate guu, i.e. the inverse of gij
guu = inv(gdd)
guu
```

- ▷ Click here to run this script.

EIGENMATH output:

$$Z = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -r\sin(\theta) & r\cos(\theta) \end{bmatrix}$$

$$g_{dd} = \begin{bmatrix} \cos(\theta)^2 + \sin(\theta)^2 & 0 \\ 0 & r^2\cos(\theta)^2 + r^2\sin(\theta)^2 \end{bmatrix}$$

$$g_{dd} = \begin{bmatrix} 1 & 0 \\ 0 & r^2 \end{bmatrix}$$

$$g_{uu} = \begin{bmatrix} 1 & 0 \\ 0 & \frac{1}{r^2} \end{bmatrix}$$

**Example 15.** (The metric tensor $g$ in cylindrical coordinates) The Euclidean metric in the cylinder coordinate systems in $\mathbb{R}^3$ can be calculated as follows, cf. [1, p.66, p.179]. Align a background Cartesian grid $(x, y, z)$ with the cylindrical coordinates $(r, \theta, z)$. The position vector $R$, pointing to a position on the cylinder, is then given by

$$R(r, \theta, z) = (r\cos(\theta), r\sin(\theta), z)$$

The basis vectors $\mathbf{Z}_i$ are the tangential vectors on the three coordinate lines, i.e. the covariant basis is obtained by partial differentiation of $R$ with respect to $r, \theta$ and $z$:

$$\mathbf{Z}_r = \frac{\partial R}{\partial r}$$

$$\mathbf{Z}_\theta = \frac{\partial R}{\partial \theta}$$

$$\mathbf{Z}_z = \frac{\partial R}{\partial z}$$

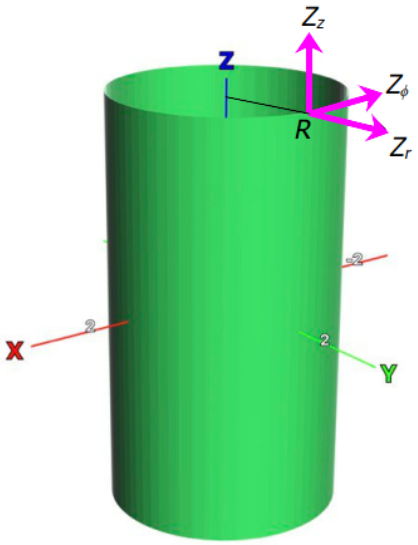



Figure 5: $R(r = 1, \theta = \pi/2, z = 2)$: position vector in $\mathbb{R}^3$.
$\partial_r$: 1st covariant basis vector $Z_r$ at $R$.
$\partial_\theta$: 2nd covariant basis vector $Z_\theta$ at $R$.
$\partial_z$: 3rd covariant basis vector $Z_z$ at $R$.

We calculate the covariant basis and the metric using EIGENMATH.

```
# EIGENMATH
-- we introduce cylindrical coordinates in R^2 by

R(r,theta,z) = (r cos(theta),r sin(theta),z)
R

-- the covariant basis in cylindrical coordinates
Z = d(R,(r,theta,z))   -- basis vectors are in the columns
Z = transpose(Z)       -- therefore we have to transpose
Z
Z[1]
Z[2]
Z[3]

-- metric tensor in cylindrical coordinates
gdd = zero(3,3)
for(i,1,3,
   for(j,1,3,
             gdd[i,j]= dot(Z[i],Z[j]) ))

gdd
gdd = simplify(gdd)
gdd

-- Calculate guu, i.e. the inverse of gij

guu = inv(gdd)
guu
```

- ▷ Click here to run this script.

EIGENMATH output:

$$g_{dd} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & r^2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$g_{uu} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1}{r^2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

## 6.3 The Contravariant Basis

With the help of the (covariant) metric tensor $g = g_{ij}$ it is possible to define the so-called *contravariant* basis $\mathbf{Z}^i$. This object plays a big role in the tensor oriented Differential Geometry. In particular, it allows resp. facilitates the calculation of the CHRISTOFFEL tensor.

### 6.3.1 Definition: contravariant Basis

Let $\mathbf{Z}_i$ be the covariant basis and $g_{ij}$ be the covariant metric tensor with corresponding contravariant metric tensor $g^{ij} = (g_{ij})^{-1}$. Then we define the $i$th contravariant basis vector $\mathbf{Z}^i$ by the linear combination, cf. [1, p.58]

$$\mathbf{Z}^i := g^{ij} \cdot \mathbf{Z}_i \overset{Einstein}{=} \sum_{i=1}^{n} g^{ij} * \mathbf{Z}_i$$

### 6.3.2 Examples: contravariant Basis

We test our definition on two of the examples above.

**Example 16.** (polar coordinates)

```
# EIGENMATH
-- ff from example 15
-- calculate the contravariant basis Zu ('Z.upper i')
do(Zu = zero(2,2),
   for(i,1,2,  Zu[i] = sum(j,1,2, guu[i,j]*Z[j])),
   Zu)

Zu[1]
Zu[2]
simplify(abs(Zu[2]))
-- compare
Z[1]
Z[2]
```

- ▷ Click here to run this script.

EIGENMATH output:

$$\begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}$$

$$\begin{bmatrix} -\frac{\sin(\theta)}{r} \\ \frac{\cos(\theta)}{r} \end{bmatrix}$$

$$\frac{1}{r}$$

**Remark.** The 1st contravariant basis vector $\mathbf{Z}^1$ in polar coordinates equals the first covariant basis vector $\mathbf{Z}_i$, whereas the 2nd contravariant basis vector $\mathbf{Z}^2$ is colinear with the second covariant basis vector and has length $|\mathbf{Z}^2| = 1/r$.

**Example 17.** (cylindrical coordinates)

```
# EIGENMATH
-- ff from example 16
-- calculate the contravariant basis Zu ('Z.upper')
do(Zu = zero(3,3),
   for(i,1,3,  Zu[i] = sum(j,1,3, guu[i,j]*Z[j])),
   Zu)

Zu
```

- ▷ Click here to run the code.

EIGENMATH output:

$$Z_u = \begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\frac{\sin(\theta)}{r} & \frac{\cos(\theta)}{r} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

**Remark.** The only difference is to be seen in the 2nd contravariant basis vector $\mathbf{Z}^2$: this vector is colinear with the 2nd covariant basis vector, but has length $|\mathbf{Z}^2| = 1/r$.

**Exercise 25.** Calculate the contravariant basis vectors for the affine coordinate system of example 13.

# 7 The Levi-Civita Tensor

The Levi-Civita tensor (symbol) acts like a metric and is often used to define tensorial objects, cf. [10, p. 340] or [9, p.50] for an application in the Nambu mechanics. It may per example also be used to define the curl operator or the cross product, cf. [1, p.148]. A heavy use is in the operation of lowering and *raising the index* in the so-called index juggling.

## 7.1 Definition

We follow Grinfeld [1, p.148].

a. Let $Z$ denote the determinant of the (covariant) metric tensor $g = g_{ij}$, i.e.

$$Z := det(g)$$

b. The square root of $Z$ is called the *volume element* $V$ of the metric, i.e.

$$V := \sqrt{det(g)}$$

c. The Levi-Civita tensor (symbols) $\epsilon_{ijk}$ resp. $\mathrm{E}^{ijk}$ are defined as the volume scaled permutation tensor $e_{ijk}$, i.e.

$$\begin{aligned} \epsilon_{ijk} &:= V \cdot e_{ijk} \\ \mathrm{E}^{ijk} &:= \frac{e^{ijk}}{V} \end{aligned}$$

**Remark.** We have

| LEXICON | *Math* | Eigenmath |
|---|---|---|
| *metric tensor* $g$: | $g_{ij}$ | `g` or `gdd` |
| *inverse of metric tensor* $g^{-1}$: | $g^{ij}$ | `ginv` or `guu` |
| *volume element* $V$: | $\sqrt{det(g)}$ | `sqrt(det(gdd))` |
| Levi-Civita *tensor* $\epsilon_{ijk}$: | $\sqrt{det(g)} \cdot e_{ijk}$ | `epsilon` |
| Levi-Civita *tensor* $\mathrm{E}^{ijk}$: | $1/\sqrt{det(g)} \cdot e^{ijk}$ | `Epsilon` |

In Eigenmath's output window the small Greek letter *epsilon* $\varepsilon$ is printed as $\epsilon$ and the big Greek letter *Epsilon* E is printed as E, not as $E$.

## 7.2 Implementation and Examples

**Example 18.** The LEVI-CIVITA tensor in an affine coordinate systems in $\mathbb{R}^2$ can be calculated by EIGENMATH as follows, cf. [1, p. 148].

```
# EIGENMATH
-- metric tensor in affine coordinates, see example 14
gdd = zero(2,2)
      gdd[1,1]= 4
      gdd[1,2]= 1
      gdd[2,1]= 1
      gdd[2,2]= 1

-- volume element in affine coordinates
Z = det(gdd)
Z

V = sqrt(Z)
V

-- permutation tensor eij in R^2 as table
eij = zero(2,2)
      for(i,1,2, for(j,1,2,  eij[i,j]=j-i ))
eij

-- LEVI-CIVITA tensor epsilon
epsilon = V * eij
epsilon

Epsilon = 1/V * eij
Epsilon
```

- ▷ Click here to run this script.

EIGENMATH output:

$$Z = 3$$

$$V = 3^{1/2}$$

$$e_{ij} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

$$\varepsilon = \begin{bmatrix} 0 & 3^{1/2} \\ -3^{1/2} & 0 \end{bmatrix}$$

$$\mathrm{E} = \begin{bmatrix} 0 & \frac{1}{3^{1/2}} \\ -\frac{1}{3^{1/2}} & 0 \end{bmatrix}$$

**Example 19.** The LEVI-CIVITA tensor in polar coordinates in $\mathbb{R}^2$ is calculated by EIGENMATH using the results of example 15.

```
# EIGENMATH
-- metric tensor in polar coordinates
gdd = zero(2,2)
      gdd[1,1]= 1
      gdd[2,2]= r^2

-- volume element in affine coordinates
Z = det(gdd)
Z

V = sqrt(Z)
V

-- permutation tensor eij in R^2 as table
eij = zero(2,2)
      for(i,1,2, for(j,1,2,  eij[i,j]=j-i ))
eij

-- LEVI-CIVITA tensor epsilon
epsilon = V * eij
epsilon

Epsilon = 1/V * eij
Epsilon
```

• ▷ Click here to run this script.

EIGENMATH output:

$$Z = r^2$$

$$V = r$$

$$e_{ij} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

$$\varepsilon = \begin{bmatrix} 0 & r \\ -r & 0 \end{bmatrix}$$

$$\mathrm{E} = \begin{bmatrix} 0 & \frac{1}{r} \\ -\frac{1}{r} & 0 \end{bmatrix}$$

**Exercise 26.** Calculate the LEVI-CIVITA tensor in cylindrical coordinates in $\mathbb{R}^3$ by EIGENMATH using the results of example 16.

# 8 The CHRISTOFFEL Tensor

The CHRISTOFFEL tensor (symbol) plays a dominant role in Differential Geometry and General Relativity, where we define the RIEMANN curvature tensor $R^a_{bcd}$, the RICCI tensor $R_{bd}$ and the EINSTEIN curvature scalar $R$ with the help of the CHRISTOFFEL tensor of 1st and 2nd kind. Then it is possible to calculate curvature variants of surfaces[7] in space. We use the results in [1, p.69] or [9, p.62] to implement the CHRISTOFFEL tensors in EIGENMATH.

## 8.1 Definition

Let $g = g_{ij}$ be a metric tensor with inverse tensor $gu = g^{ij}$ in the Cartesian coordinate system $(x_1, x_2, x_3)$ in $\mathbb{R}^3$.

**a.** The CHRISTOFFEL *tensor of 2nd kind* is defined by (EINSTEIN summation convention!)

$$\Gamma^k_{ij} := \frac{1}{2} \cdot g^{km} \cdot (g_{mi,j} + g_{mj,i} - g_{ij,m})$$

where

$$g_{mi,j} := \frac{\partial g_{mi}}{\partial x_j}$$

denotes the partial derivative of $g_{mi}$ with respect to the coordinate $x_j$. Expanding the EINSTEIN summation convention we write equation a. more explicit as

$$\Gamma^k_{ij} = \frac{1}{2} \cdot \sum_{m=1}^{3} g^{km} \cdot (g_{mi,j} + g_{mj,i} - g_{ij,m})$$

**b.** The CHRISTOFFEL *tensor of 2nd kind* is defined by (EINSTEIN summation convention!)

$$\Gamma_{i,jk} := g_{li} \cdot \Gamma^l{}_{jk}$$

This last definition is an example of lowering the index $l$ by multiplying with the metric tensor $g$ ('index juggling').

**c.** If we denote the partial derivative with respect to the coordinate $x_j$ in short by $\partial_i$, we may write formula a. in better memorizable shape, cf. [4, p.91]:

$$\Gamma^k_{ij} = \frac{1}{2} \cdot \sum_{m=1}^{3} g^{km} \cdot (\partial_j g_{mi} + \partial_i g_{mj} - \partial_m g_{ij})$$

Implementing formulas a. and b. in EIGENMATH should enhance a clear understanding of these constructs.

[7] This will be done in the next script in this series with the title *Elementary Differential Geometry of Surfaces*.

**Remark.** We have

| LEXICON | *Math* | EIGENMATH |
|---|---|---|
| *metric tensor g*: | $g_{ij}$ | `g` or `gdd` |
| *inverse of metric tensor* $g^{-1}$: | $g^{ij}$ | `ginv` or `guu` |
| CHRISTOFFEL *tensor of 1st kind*: | $\Gamma^k_{ij}$ | `Gammaudd` |
| CHRISTOFFEL *tensor of 2nd kind*: | $\Gamma_{i,jk}$ | `Gammaddd` |

In EIGENMATH's output window the $\Gamma^k_{ij}$ is printed as $\Gamma_{udd}$.

## 8.2 Implementation and Examples

We now demonstrate two implementations of the CHRISTOFFEL tensors: one closely oriented on formula **a**. and its term structure and the other using clever compact tensor techniques such as contract and index juggling, with was given by G. WEIGT. We test our code in affine, polar and cylindrical coordinates in the plane $\mathbb{R}^2$ resp. in space $\mathbb{R}^3$.

### 8.2.1 *Calculation of the* CHRISTOFFEL *tensor using standard techniques*

**Example 20.** The CHRISTOFFEL tensor in *affine coordinates* in $\mathbb{R}^2$ is calculated using results of example 15. Let's step through all coefficients of the CHRISTOFFEL tensor using a for-loop.

```
# EIGENMATH
-- metric tensor in affine coordinates
gdd = zero(2,2)
      gdd[1,1]= 4
      gdd[1,2]= 1
      gdd[2,1]= 1
      gdd[2,2]= 1

--  use Cartesian coordinates
X = (x,y)

-- calculate inverse guu to gdd
guu = inv(gdd)
guu

-- calculate CHRISTOFFEL tensor
Gamma=zero(2,2,2)    -- tensor shape
for( k,1,2,          -- fill Gamma
    for( l,1,2,
        for( m,1,2,
             Gamma[k,l,m] =                    -- (1)
                sum(i,1,2, guu[k,i]/2 *        -- (2)
                      ( d( gdd[m,i], X[l]) +   -- (3)
```

```
                        d( gdd[l,i], X[m]) -
                        d( gdd[m,l], X[i]) )
                ))))

Gamma           -- (4)
Gamma[1,2,2]    -- (5)
Gamma[2,1,2]
```

- ▷ Click here to run this script.

EIGENMATH output:

$$g_{uu} = \begin{bmatrix} \frac{1}{3} & -\frac{1}{3} \\ -\frac{1}{3} & \frac{4}{3} \end{bmatrix}$$

$$\Gamma = \begin{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{bmatrix}$$

$$0$$

$$0$$

**Comment.** In (1) we give `Gamma[k,l,m]` $= \Gamma^k_{lm}$ the value in accordance with formula 7.1.a. By the way, we respect the EINSTEIN summation convention by explicit summing the three partial derivatives up. The resulting fully filled tensor is outputted in (4). (5) verifies that all components of the affine CHRISTOFFEL tensor tensor $\Gamma$ vanishes at all points. Later we will argue, that therefore this space is 'flat'.

**Example 21.** Calculate the CHRISTOFFEL tensor in a polar coordinate system in $\mathbb{R}^2$ using results of example 16.

```
# EIGENMATH
-- metric tensor in polar coordinates
gdd = zero(2,2)
      gdd[1,1]= 1
      gdd[2,2]= r^2

-- use polar coordinates
X = (r,theta)
X
```

```
-- calculate guu = (gdd)^{-1}
guu = inv(gdd)
guu

-- calculate Gamma using a for-loop
Gamma=zero(2,2,2)        -- tensor of type (2,2,2)

for( k,1,2,
    for( l,1,2,
        for( m,1,2,
             Gamma[k,l,m] =
                sum(i,1,2, guu[k,i]/2 *
                      ( d(gdd[m,i],X[l]) +
                        d(gdd[l,i],X[m]) -
                        d(gdd[m,l],X[i]) )
                ))))

Gamma           --(1)
Gamma[1,2,2]    --(2)
Gamma[2,1,2]
```

- ▷ Click here to run this script.

EIGENMATH output:

$$\Gamma = \begin{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & -r \end{bmatrix} \\ \begin{bmatrix} 0 & \frac{1}{r} \\ \frac{1}{r} & 0 \end{bmatrix} \end{bmatrix}$$

$$-r$$

$$\frac{1}{r}$$

**Comment.** In (1) we see that the fully filled CHRISTOFFEL tensor $\Gamma$ has non-zero komponents. (6) picks two of these non-zero components of $\Gamma$, namely $\Gamma^1_{22}$ and $\Gamma^2_{12}$.

**Example 22.** Calculate the CHRISTOFFEL tensor in cylindrical coordinates in $\mathbb{R}^3$ using results of example 17.

```
# EIGENMATH
-- metric tensor in cylindrical coordinates
gdd = zero(3,3)
      gdd[1,1]= 1
      gdd[2,2]= r^2
      gdd[3,3]= 1

gdd

-- X is cylindrical coordinates, t means theta
X = (r,t,z)
X

-- calculate guu, i.e. the inverse of gij
guu = inv(gdd)
guu

n=3   -- we are now in a 3-dimensional world
Gamma=zero(n,n,n)   -- tensor of type (3,3,3)

for( k,1,n,
    for( l,1,n,
        for( m,1,n,
             Gamma[k,l,m] =
                sum(i,1,2, guu[k,i]/2 *
                      ( d(gdd[m,i],X[l]) +
                        d(gdd[l,i],X[m]) -
                        d(gdd[m,l],X[i]) )
                ))))

Gamma            --(1)
Gamma[1,2,2]     --(2)
Gamma[2,1,2]
Gamma[2,2,1]
```

- ▷ Click here to run this script.

**Comment.** Here we have metric $g$ and CHRISTOFFEL tensor $\Gamma$ of 3 dimensions. Therefore the CHRISTOFFEL tensor is stacked with three 3-by-3 matrices. Only the first two $\Gamma^1_{\cdot\cdot}$ and $\Gamma^2_{\cdot\cdot}$ has non-zero entries. We pick three of them in (2).

EIGENMATH output:

$$\Gamma = \begin{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -r & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & \frac{1}{r} & 0 \\ \frac{1}{r} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{bmatrix}$$

$$-r$$

$$\frac{1}{r}$$

$$\frac{1}{r}$$

### 8.2.2 *Calculation of the* CHRISTOFFEL *tensor using tensor techniques*

**Example 23.** Calculate the CHRISTOFFEL tensor in affine coordinates in $\mathbb{R}^2$ using the results of example 15 and tensor techniques following G. WEIGT.

```
# EIGENMATH
-- metric tensor in affine coordinates
gdd = zero(2,2)
      gdd[1,1]= 4
      gdd[1,2]= 1
      gdd[2,1]= 1
      gdd[2,2]= 1

-- use Cartesian coordinates
X = (x,y)
X

-- calculate guu
guu = inv(gdd)
guu

-- calculate all partial derivatives w.r.t. x and y
gddd = d( gdd, X)
gddd                -- a tensor with 2 Jacobi matrices
```

```
-- Calculate so-called connection coefficients aka the
-- CHRISTOFFEL symbol of 1st kind along formula 7.1.b.

-- Gamma    = 1/2 (g     + g     - g    )
--      abc         ab,c    ac,b    bc,a
--      123         12 3    13 2    23 1
--

Gammaddd = 1/2*( gddd +                      -- Gab,c: correct order
                 transpose(gddd,2,3) -       -- Gac,b: transpose c and b
                 transpose(gddd,2,3,1,2))    -- Gbc,a: transpose c and a,
                                             --        then b and a
Gammaddd         --(1)

-- Raise first index a ('index juggling') using guu, see 7.1.b
--
--      a      au
-- Gamma    = g   Gamma              (2)
--       bc            ubc

Gammaudd = dot(guu,Gammaddd)        --(3)
Gammaudd                            --(4) CHRISTOFFEL tensor of 2nd kind
```

• ▷ Click here to run this script.

EIGENMATH output:

$$\Gamma_{ddd} = \begin{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{bmatrix} \qquad \Gamma_{udd} = \begin{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \end{bmatrix}$$

**Comment.** In (1) we give `Gammaddd`$= \Gamma_{abc}$ the value in accordance with formula 7.1.b. In formula (2) index $u$ is both upper and lower to be seen, ergo there is an implicit summation over $u$ (' contraction') w.r.t. the EINSTEIN summation convention. This is done in (3). The resulting tensor `Gammaudd`$= \Gamma^a_{bc}$ is outputted in (4). It is the affine CHRISTOFFEL tensor tensor $\Gamma$ with the same result as in example 21.

**Example 24.** Calculate the CHRISTOFFEL tensor in polar coordinates in $\mathbb{R}^2$ using the results of example 16 and tensor techniques following G. WEIGT.

```
# EIGENMATH

-- metric tensor in polar coordinates
gdd = zero(2,2)
      gdd[1,1]= 1
      gdd[2,2]= r^2

-- use polar coordinates
X = (r,theta)

-- calculate guu
guu = inv(gdd)

-- calculate all partial derivations w.r.t. r, theta
gddd = d( gdd, X)
gddd                 --(1)

-- calculate CHRISTOFFEL tensor 1st kind
-- see explanation in previous example 24

Gammaddd = 1/2*(gddd +
                transpose(gddd,2,3) -
                transpose(gddd,2,3,1,2))

Gammaddd            --(2)

-- raise first index using guu, i.e. .ddd -> .udd
Gammaudd = dot(guu,Gammaddd) -- (3)
Gammaudd
```

- ▷ Click here to run this script.

EIGENMATH output:

$$g_{ddd} = \begin{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \\ 2r & 0 \end{bmatrix} \end{bmatrix} \qquad \Gamma_{ddd} = \begin{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & -r \end{bmatrix} \\ \begin{bmatrix} 0 & r \\ r & 0 \end{bmatrix} \end{bmatrix} \qquad \Gamma_{udd} = \begin{bmatrix} \begin{bmatrix} 0 & 0 \\ 0 & -r \end{bmatrix} \\ \begin{bmatrix} 0 & \frac{1}{r} \\ \frac{1}{r} & 0 \end{bmatrix} \end{bmatrix}$$

**Comment.** In (1) we collect all partial derivatives w.r.t. $r$ and $\theta$, we then calculate `Gammaddd`, which is the CHRISTOFFEL tensor $\Gamma_{abc}$ of 1st kind. Raising the first index $u$ (see previous example) results in an implicit summation over $u$ in (3). The resulting CHRISTOFFEL tensor `Gammaudd`$= \Gamma^a_{bc}$ of 2nd kind is outputted giving the same result as in example 22.

**Example 25.** Calculate the CHRISTOFFEL tensor in cylindrical coordinates in $\mathbb{R}^3$ using results of example 17 and tensor techniques following G. WEIGT.

```
# EIGENMATH
-- metric tensor in cylindrical coordinates
gdd = zero(3,3)
      gdd[1,1]= 1
      gdd[2,2]= r^2
      gdd[3,3]= 1

-- use cylindrical coordinates
X = (r,t,z)

-- Calculate guu, i.e. the inverse of gij
guu = inv(gdd)

-- calculate all partial derivations w.r.t. r,t and z
gddd = d( gdd, X)
gddd

-- calculate CHRISTOFFEL symbol of 1st kind
Gammaddd = 1/2*( gddd                    +
                 transpose(gddd,2,3)     -
                 transpose(gddd,2,3,1,2))

Gammaddd

-- Raise first index
Gammaudd = dot( guu, Gammaddd )
Gammaudd
```

- ▷ Click here to run this script.

EIGENMATH output:

$$g_{ddd} = \begin{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 2r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{bmatrix} \qquad \Gamma_{ddd} = \begin{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -r & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & r & 0 \\ r & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{bmatrix} \qquad \Gamma_{udd} = \begin{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -r & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & \frac{1}{r} & 0 \\ \frac{1}{r} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \end{bmatrix}$$

**Comment.** We repeat the same tensorial calculation as in the examples before. In contrast to the classic solution using for-loops the higher dimension of $\Gamma$ is automatically detected, no special dimension marker like '$n = 3$' is necessary: the dimension of CHRISTOFFEL tensor $\Gamma^{\cdot}_{\cdot\cdot}$ is taken from the declaration $X = (r, t, z)$, which is a 3-vector.

**Exercise 27.** Calculate the CHRISTOFFEL tensor $\Gamma_{abc}$ of 1st kind for the last three examples using the classical way via for-loops. Verify your calculation with a check of the examples 24, 25 and 26.

♡

This ends our journey to the first elements of Tensor Calculus. See you again doing the first steps into the *Elementary Differential Geometry of Surfaces*.

# 9 Appendix: source code of tensorBox

This is a collection of the relevant definitions from the booklet. This file should be loaded to do calculations with tensors using EIGENMATH.

```
################ (2023) Dr.W.G. Lindner,  Leichlingen DE
###    tensorBox       Tensor box
################       for calculations with tensors

-- KRONECKER delta function d(i,j)
delta(i,j) = test( i=j, 1, 0)

- KRONECKER delta 2-tensor dij
deltaij = zero(2,2)
   deltaij[1,1] = 1
   deltaij[2,2] = 1

-- 2D permutation tensor als FUNCTION
Eij(i,j) = j-i
Eij(1,2)
Eij(1,1)

-- 2D permutation tensor eij as TABLE
eij = zero(2,2)
for(i,1,2, for(j,1,2,  eij[i,j]=j-i ))

-- 3D permutation tensor as FUNCTION
Eijk(i,j,k) = test(
   or( (i,j,k)==(1,2,3), (i,j,k)==(2,3,1), (i,j,k)==(3,1,2)), +1,
   or( (i,j,k)==(3,2,1), (i,j,k)==(1,3,2), (i,j,k)==(2,1,3)), -1,
   or( i==j, j==k, k==i), 0 )

-- 3D permutation tensor as TABLE
eijk = zero(3,3,3)
  eijk[1,2,3] =  1
  eijk[2,3,1] =  1
  eijk[3,1,2] =  1
  eijk[3,2,1] = -1
  eijk[1,3,2] = -1
  eijk[2,1,3] = -1


-- 4D permutation tensor as FUNCTION
Eijkl(i,j,k,l) = (i-j)*(i-k)*(i-l)*(j-k)*(j-l)*(k-l)/12

-- 4D permutation tensor as TABLE
```

```
eijkl=zero(4,4,4,4)
for(i,1,4, for(j,1,4, for(k,1,4, for(l,1,4,
        eijkl[i,j,k,l] = (i-j)*(i-k)*(i-l)*(j-k)*(j-l)*(k-l)/12 ) )))

-- metric tensor g in 2D, an example
gdd = zero(2,2)
      gdd[1,1]= 4
      gdd[1,2]= 1
      gdd[2,1]= 1
      gdd[2,2]= 1

-- inverse guu to gdd
guu = inv(gdd)

-- volume element
Z = det(gdd)
V = sqrt(Z)

-- LEVI-CIVITA tensor epsilon in 2D
epsilon = V * eij

Epsilon = 1/V * eij

-- 2D/3D/nD CHRISTOFFEL tensor 2nd kind
-- specify dimension n = 2,3,..
n=3
Gamma=zero(n,n,n)
for( k,1,n,
    for( l,1,n,
        for( m,1,n,
             Gamma[k,l,m] =
                sum(i,1,2, guu[k,i]/2 *
                      ( d(gdd[m,i],X[l]) +
                        d(gdd[l,i],X[m]) -
                        d(gdd[m,l],X[i]) )
                ))))
```

- ▷ Click here to get the code.

# References

[1] Grinfeld, P. (2013): *Introduction to Tensor Analysis and the Calculus of Moving Surfaces.* New York: Springer.

[2] Lipschutz, M. M. (1969): *Differential Geometry.*
New York: McGraw-Hill (Schaums' Outline Series)

[3] Marsden, J. & Weinstein, A. ($^2$1985): *Calculus III.* New York: Springer.

[4] Oloff, R. ($^6$2018): *Geometrie der Raumzeit.* [engl.: Geometry of Spacetime.]
Berlin: Springer Spectrum.

[5] Seeburger, P. (2018): *CalcPlot3D.*
url: `https://c3d.libretexts.org/CalcPlot3D/index.html`

[6] Sochi, T. (2017): *Tensor Calculus Made Simple.*
?: CreativeSpace. ISBN 9781541013636.

[7] Sochi, T. (2017): *Principles of Tensor Calculus.*
?: CreativeSpace. ISBN 9781974401390.

[8] Spiegel, M. R. (1963): *Advanced Calculus.*
New York: McGraw-Hill (Schaums' Outline Series)

[9] Steeb, W.-H & Lewien, D. (1991): *Algorithms and Computation with* Reduce.
Mannheim: Bibliographisches Institut.

[10] Steeb, W.-H & Shi, T. K. (1998): *Symbolic C++: An Introduction to Computer Algebra Using Object-Oriented Programming.*
Singapore: Springer.

[11] Steeb, W.-H (1991): *Kronecker Product of Matrices and Applications.*
Mannheim: Bibliographisches Institut.

[12] Steeb, W.-H, Hardy, Y. (2019): *Kronecker Product.* In: *Matrix Calculus, Kronecker Product and Tensor Product.*
url: `https://www.worldscientific.com/doi/pdf/10.1142/9789811202520_0002`

[13] Weigt, G. (2021): Eigenmath *Homepage.*
url: `https://georgeweigt.github.io`

[14] Weigt, G. (2021): Eigenmath *online.*
url: `https://georgeweigt.github.io/eigenmath-demo.html`

[15] Weigt, G. (2021): *The Schwarzschild Metric.*
url: `https://georgeweigt.github.io/schwarzschild-metric.html`

[16] WikipediA: Kronecker Product.
url: `https://en.wikipedia.org/wiki/Kronecker_product`

[17] WikipediA: Levi-Civita-Symbol.
url: `https://de.wikipedia.org/wiki/Levi-Civita-Symbol`

[18] WikipediA: Metric Tensor.
url: `https://en.wikipedia.org/wiki/Metric_tensor`

[19] WikipediA: Outer Product.
url: `https://en.wikipedia.org/wiki/Outer_product`

[20] WikipediA: Tensor Contraction.
url: `https://en.wikipedia.org/wiki/Tensor_contraction`