(Anti)Symmetrization of Tensors Levi-Civita Symbol

Permutations

The Function Permutations[list] gives all possible permutations of the elements in the list

In[ ]:=
```
Permutations[{1, 2, 3}]
```
Out[ ]=
```
{{1, 2, 3}, {1, 3, 2}, {2, 1, 3}, {2, 3, 1}, {3, 1, 2}, {3, 2, 1}}
```

In[ ]:=
```
Permutations[{μ, ν, ρ}]
```
Out[ ]=
```
{{μ, ν, ρ}, {μ, ρ, ν}, {ν, μ, ρ}, {ν, ρ, μ}, {ρ, μ, ν}, {ρ, ν, μ}}
```

Signature[list] gives the signature of the permutation needed to place the elements of *list* in canonical order: (see the function Sort for definition of canonical order)

In[ ]:=
```
Print[Signature[{2, 3, 1}], " ", Signature[{2, 1, 3}]]
```
```
1 -1
```

In[ ]:=
```
Print[Signature[{μ, ν, ρ}], " ", Signature[{μ, ρ, ν}]]
```
```
1 -1
```

Signature must be applied to all elements of the list to give their signature.
Use Map[perm3,Signature] or the equivalent Signature /@ perm3

In[ ]:=
```
perm3 = Permutations[{1, 2, 3}];
Signature /@ perm3
```
Out[ ]=
```
{1, -1, -1, 1, 1, -1}
```

Show the permutations together with their Signature:

In[ ]:=
```
Table[{perm3[[i]], Signature[perm3[[i]]]}, {i, Length[perm3]}] // Column
```

Out[ ]=
```
{{1, 2, 3}, 1}
{{1, 3, 2}, -1}
{{2, 1, 3}, -1}
{{2, 3, 1}, 1}
{{3, 1, 2}, 1}
{{3, 2, 1}, -1}
```

Test whether particular permutations are even:

In[ ]:=
```
{Signature[{1, 2, 3}] == 1, Signature[{1, 3, 2}] == 1}
```

Out[ ]=
```
{True, False}
```

(Signature[#] ==1) &
is an unnamed (pure) function. Like any other function f, which takes an argument x and evaluates to f[x], it takes an argument
(Signature[#] ==1) & [perm]
The argument perm is substituted at all positions in the expression that has a #, and evaluates to
(Signature[perm]==1)
which is True, or False:

```
(Signature[#] == 1) & [{1, 2, 3}]
```

Out[ ]=
```
True
```

We can map this function on all permutations:

In[ ]:=
```
(Signature[#] == 1) & /@ perm3
```

Out[ ]=
```
{True, False, False, True, True, False}
```

Select even permutations:
Select[list,crit] picks out elements of the list for which crit[el] is True.

Then, Select can be used to to apply the pure function (Signature[#] == 1)& to all elements in the list, and select the ones which return True:

In[ ]:=
```
Select[perm3, (Signature[#] == 1) & ]
```

Out[ ]=
```
{{1, 2, 3}, {2, 3, 1}, {3, 1, 2}}
```

Odd permutations:

In[ ]:=
```
Select[perm3, (Signature[#] == -1) & ]
```
Out[ ]=
```
{{1, 3, 2}, {2, 1, 3}, {3, 2, 1}}
```

(Anti)symmetrization

Define a permutation of all indices of a tensor:

In[ ]:=
```
perm3 = Permutations[{a1, a2, a3}]
```
Out[ ]=
```
{{a1, a2, a3}, {a1, a3, a2}, {a2, a1, a3}, {a2, a3, a1}, {a3, a1, a2}, {a3, a2, a1}}
```

The function Subscript[s,a] returns the object s_a:

In[ ]:=
```
Subscript[s, a1]
```
Out[ ]=
s_{a1}

We can add more subscripts:

In[ ]:=
```
Subscript[s, a1, a2, a3]
```
Out[ ]=
$s_{a1,a2,a3}$

Construct all permutations of $s_{a1,a2,a3}$ by applying Subscript[s,#[[1]],#[[2]],#[[3]]]& on a list {a1,a2,a3} of 3 elements.
#[[1]] is the first element of the argument of the function, and it will be replaced by a1. Similarly for #[[2]] and #[[3]].

In[ ]:=
```
Subscript[s, #[[1]], #[[2]], #[[3]]] &[{a1, a2, a3}]
```
Out[ ]=
$s_{a1,a2,a3}$

Since perm3 is a list of permuted objects, Subscript must be applied on each member of the list:

In[ ]:=
```
(Subscript[s, #[[1]], #[[2]], #[[3]]]) & /@ perm3
```
Out[ ]=
$\{s_{a1,a2,a3}, s_{a1,a3,a2}, s_{a2,a1,a3}, s_{a2,a3,a1}, s_{a3,a1,a2}, s_{a3,a2,a1}\}$

We can multiply with the signature of the permutation:

In[ ]:=
```
signed3 = (Signature[#] Subscript[s, #[[1]], #[[2]], #[[3]]]) & /@ perm3
```
Out[ ]=
$\{s_{a1,a2,a3}, -s_{a1,a3,a2}, -s_{a2,a1,a3}, s_{a2,a3,a1}, s_{a3,a1,a2}, -s_{a3,a2,a1}\}$

We can obtain the sum of those terms, by applying Plus on the list.

(the function Apply works by replacing the head List by the head Plus)

In[ ]:=
```
Apply[Plus, signed3]
```

Out[ ]= $s_{a1,a2,a3} - s_{a1,a3,a2} - s_{a2,a1,a3} + s_{a2,a3,a1} + s_{a3,a1,a2} - s_{a3,a2,a1}$

or equivalently:

In[ ]:=
```
Plus @@ signed3
```

Out[ ]= $s_{a1,a2,a3} - s_{a1,a3,a2} - s_{a2,a1,a3} + s_{a2,a3,a1} + s_{a3,a1,a2} - s_{a3,a2,a1}$

Then we can write down the expression that gives the antisymmetrization $s_{[a1\,a2\,a3]}$

In[ ]:=
```
(1 / 3!) Plus @@ signed3
```

Out[ ]= $\frac{1}{6}\left(s_{a1,a2,a3} - s_{a1,a3,a2} - s_{a2,a1,a3} + s_{a2,a3,a1} + s_{a3,a1,a2} - s_{a3,a2,a1}\right)$

or, directly:

In[ ]:=
```
Plus @@
  ((Signature[#] Subscript[s, #[[1]], #[[2]], #[[3]]]) & /@ Permutations[{a1, a2, a3}]) (1 / 3!)
```

Out[ ]= $\frac{1}{6}\left(s_{a1,a2,a3} - s_{a1,a3,a2} - s_{a2,a1,a3} + s_{a2,a3,a1} + s_{a3,a1,a2} - s_{a3,a2,a1}\right)$

Higher order tensor:

In[ ]:=
```
perm4 = Permutations[{a1, a2, a3, a4}];
(1 / Length[perm4])
 Apply[Plus, (Signature[#] Subscript[s, #[[1]], #[[2]], #[[3]], #[[4]]]) & /@ perm4 ]
```

Out[ ]=
$$\frac{1}{24}\left(s_{a1,a2,a3,a4} - s_{a1,a2,a4,a3} - s_{a1,a3,a2,a4} + s_{a1,a3,a4,a2} + s_{a1,a4,a2,a3} - s_{a1,a4,a3,a2} - s_{a2,a1,a3,a4} + s_{a2,a1,a4,a3} + s_{a2,a3,a1,a4} - s_{a2,a3,a4,a1} - s_{a2,a4,a1,a3} + s_{a2,a4,a3,a1} + s_{a3,a1,a2,a4} - s_{a3,a1,a4,a2} - s_{a3,a2,a1,a4} + s_{a3,a2,a4,a1} + s_{a3,a4,a1,a2} - s_{a3,a4,a2,a1} - s_{a4,a1,a2,a3} + s_{a4,a1,a3,a2} + s_{a4,a2,a1,a3} - s_{a4,a2,a3,a1} - s_{a4,a3,a1,a2} + s_{a4,a3,a2,a1}\right)$$

We can symmetrize or antisymmetrize tensors using the Symmetrize function.
This is a (0,2) tensor:

In[•]:=
```
t1 = Array[A## &, {2, 2}];(*will Array[] explain later...*)
t1 // MatrixForm
```

Out[•]//MatrixForm=

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix}$$

Here, we antisymmetrize wrt to the two indices in positions 1 and 2

In[•]:=
```
Symmetrize[t1, Antisymmetric[{1, 2}]] // Normal // MatrixForm
```

Out[•]//MatrixForm=

$$\begin{pmatrix} 0 & \frac{1}{2}\left(A_{1,2} - A_{2,1}\right) \\ \frac{1}{2}\left(-A_{1,2} + A_{2,1}\right) & 0 \end{pmatrix}$$

Now we symmetrize:

In[•]:=
```
Symmetrize[t1, Symmetric[{1, 2}]] // Normal // MatrixForm
```

Out[•]//MatrixForm=

$$\begin{pmatrix} A_{1,1} & \frac{1}{2}\left(A_{1,2} + A_{2,1}\right) \\ \frac{1}{2}\left(A_{1,2} + A_{2,1}\right) & A_{2,2} \end{pmatrix}$$

A is defined as a (0,3) tensor

In[•]:=
```
t2 = Array[A## &, {3, 3, 3}];
t2 // MatrixForm
```

Out[•]//MatrixForm=

$$\begin{pmatrix} \begin{pmatrix} A_{1,1,1} \\ A_{1,1,2} \\ A_{1,1,3} \end{pmatrix} & \begin{pmatrix} A_{1,2,1} \\ A_{1,2,2} \\ A_{1,2,3} \end{pmatrix} & \begin{pmatrix} A_{1,3,1} \\ A_{1,3,2} \\ A_{1,3,3} \end{pmatrix} \\ \begin{pmatrix} A_{2,1,1} \\ A_{2,1,2} \\ A_{2,1,3} \end{pmatrix} & \begin{pmatrix} A_{2,2,1} \\ A_{2,2,2} \\ A_{2,2,3} \end{pmatrix} & \begin{pmatrix} A_{2,3,1} \\ A_{2,3,2} \\ A_{2,3,3} \end{pmatrix} \\ \begin{pmatrix} A_{3,1,1} \\ A_{3,1,2} \\ A_{3,1,3} \end{pmatrix} & \begin{pmatrix} A_{3,2,1} \\ A_{3,2,2} \\ A_{3,2,3} \end{pmatrix} & \begin{pmatrix} A_{3,3,1} \\ A_{3,3,2} \\ A_{3,3,3} \end{pmatrix} \end{pmatrix}$$

Here we antisymmetrize only indices at positions 1 and 3 to obtain $A_{[a|b|c]}$

In[ ]:=
```
Symmetrize[t2, Antisymmetric[{1, 3}]] // Normal // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} \begin{pmatrix} 0 \\ \frac{1}{2}(A_{1,1,2}-A_{2,1,1}) \\ \frac{1}{2}(A_{1,1,3}-A_{3,1,1}) \end{pmatrix} & \begin{pmatrix} 0 \\ \frac{1}{2}(A_{1,2,2}-A_{2,2,1}) \\ \frac{1}{2}(A_{1,2,3}-A_{3,2,1}) \end{pmatrix} & \begin{pmatrix} 0 \\ \frac{1}{2}(A_{1,3,2}-A_{2,3,1}) \\ \frac{1}{2}(A_{1,3,3}-A_{3,3,1}) \end{pmatrix} \\ \begin{pmatrix} \frac{1}{2}(-A_{1,1,2}+A_{2,1,1}) \\ 0 \\ \frac{1}{2}(A_{2,1,3}-A_{3,1,2}) \end{pmatrix} & \begin{pmatrix} \frac{1}{2}(-A_{1,2,2}+A_{2,2,1}) \\ 0 \\ \frac{1}{2}(A_{2,2,3}-A_{3,2,2}) \end{pmatrix} & \begin{pmatrix} \frac{1}{2}(-A_{1,3,2}+A_{2,3,1}) \\ 0 \\ \frac{1}{2}(A_{2,3,3}-A_{3,3,2}) \end{pmatrix} \\ \begin{pmatrix} \frac{1}{2}(-A_{1,1,3}+A_{3,1,1}) \\ \frac{1}{2}(-A_{2,1,3}+A_{3,1,2}) \\ 0 \end{pmatrix} & \begin{pmatrix} \frac{1}{2}(-A_{1,2,3}+A_{3,2,1}) \\ \frac{1}{2}(-A_{2,2,3}+A_{3,2,2}) \\ 0 \end{pmatrix} & \begin{pmatrix} \frac{1}{2}(-A_{1,3,3}+A_{3,3,1}) \\ \frac{1}{2}(-A_{2,3,3}+A_{3,3,2}) \\ 0 \end{pmatrix} \end{pmatrix}$$

Now compute $A_{[abc]}$:

Notice that $A \neq 0$ is not evaluated to True, so we cannot Select nonzero elements this way.

Instead we use (!MatchQ[A,0]), meaning “A does not match to 0”

In[ ]:=
```
at2 = Symmetrize[t2, Antisymmetric[{1, 2, 3}]] // Normal;
Select[Flatten[at2], (! MatchQ[#, 0]) &] // Column
```

Out[ ]=

$$\frac{1}{6}(A_{1,2,3}-A_{1,3,2}-A_{2,1,3}+A_{2,3,1}+A_{3,1,2}-A_{3,2,1})$$
$$\frac{1}{6}(-A_{1,2,3}+A_{1,3,2}+A_{2,1,3}-A_{2,3,1}-A_{3,1,2}+A_{3,2,1})$$
$$\frac{1}{6}(-A_{1,2,3}+A_{1,3,2}+A_{2,1,3}-A_{2,3,1}-A_{3,1,2}+A_{3,2,1})$$
$$\frac{1}{6}(A_{1,2,3}-A_{1,3,2}-A_{2,1,3}+A_{2,3,1}+A_{3,1,2}-A_{3,2,1})$$
$$\frac{1}{6}(A_{1,2,3}-A_{1,3,2}-A_{2,1,3}+A_{2,3,1}+A_{3,1,2}-A_{3,2,1})$$
$$\frac{1}{6}(-A_{1,2,3}+A_{1,3,2}+A_{2,1,3}-A_{2,3,1}-A_{3,1,2}+A_{3,2,1})$$

Antisymmetric[{1,2,3}] gives the same result as Antisymmetric[All]

In[ ]:=
```
at2 = Symmetrize[t2, Antisymmetric[All]] // Normal;
Select[Flatten[at2], (! MatchQ[#, 0]) &] // Column
```

Out[ ]=

$$\frac{1}{6}(A_{1,2,3}-A_{1,3,2}-A_{2,1,3}+A_{2,3,1}+A_{3,1,2}-A_{3,2,1})$$
$$\frac{1}{6}(-A_{1,2,3}+A_{1,3,2}+A_{2,1,3}-A_{2,3,1}-A_{3,1,2}+A_{3,2,1})$$
$$\frac{1}{6}(-A_{1,2,3}+A_{1,3,2}+A_{2,1,3}-A_{2,3,1}-A_{3,1,2}+A_{3,2,1})$$
$$\frac{1}{6}(A_{1,2,3}-A_{1,3,2}-A_{2,1,3}+A_{2,3,1}+A_{3,1,2}-A_{3,2,1})$$
$$\frac{1}{6}(A_{1,2,3}-A_{1,3,2}-A_{2,1,3}+A_{2,3,1}+A_{3,1,2}-A_{3,2,1})$$
$$\frac{1}{6}(-A_{1,2,3}+A_{1,3,2}+A_{2,1,3}-A_{2,3,1}-A_{3,1,2}+A_{3,2,1})$$

Levi-Civita symbol

The function Array[f,{n1,n2}] generates a nested list of n1 × n2 elements f[i1,i2]:

In[ ]:=
```
Array[f, {3, 3}] // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} f[1,1] & f[1,2] & f[1,3] \\ f[2,1] & f[2,2] & f[2,3] \\ f[3,1] & f[3,2] & f[3,3] \end{pmatrix}$$

The array $A_{ij} = i + j$

Plus is applied on {i,j}, giving Plus[i,j]=i+j

In[ ]:=
```
Array[Plus, {3, 3}] // MatrixForm
```

Out[ ]//MatrixForm=

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

Now, define a n1 × n2 × n3 array:

In[ ]:=
```
Array[f, {2, 2, 2}] // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} \begin{pmatrix} f[1,1,1] \\ f[1,1,2] \end{pmatrix} & \begin{pmatrix} f[1,2,1] \\ f[1,2,2] \end{pmatrix} \\ \begin{pmatrix} f[2,1,1] \\ f[2,1,2] \end{pmatrix} & \begin{pmatrix} f[2,2,1] \\ f[2,2,2] \end{pmatrix} \end{pmatrix}$$

In[ ]:=
```
Array[f, {2, 2, 2}]
```

Out[ ]=
```
{{{f[1, 1, 1], f[1, 1, 2]}, {f[1, 2, 1], f[1, 2, 2]}},
 {{f[2, 1, 1], f[2, 1, 2]}, {f[2, 2, 1], f[2, 2, 2]}}}
```

So if f is replaced by the pure function Signature, we can obtain the 3-d Levi-Civita Symbol.

Let’s do it in steps:

This is what the Array function gives: {i,j,k} → f[i,j,k]

In[ ]:=
```
f[1, 2, 3]
```

Out[ ]=
```
f[1, 2, 3]
```

But Signature needs to act on the list {i,j,k}. So we have to take the arguments 1,2,3 and form a list {1,2,3} on which the function will act.

We will use the symbol ##, which represents the sequence of all arguments supplied to a pure function and put it inside {} to form a list:

```
In[ ]:= f[{##}] & [1, 2, 3]
Out[ ]= f[{1, 2, 3}]
```

So, we got what we want: A function that gives the signature of *its arguments*!

```
In[ ]:= (Signature[{##}]) & [1, 2, 3]
Out[ ]= 1
```

We go ahead and apply this function on the arguments given to it by Array:

```
In[ ]:= Array[Signature[{##}] &, {3, 3, 3}] // Column
Out[ ]= {{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}}
```

This is important, so Mathematica has already a function LeviCivitaTensor:
It is a sparse array, so we apply the function Normal to obtain an ordinary list

```
In[ ]:= LeviCivitaTensor[3] // Normal  // Column
Out[ ]= {{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}}
```

Or, obtain the same using:

```
In[ ]:= LeviCivitaTensor[3, List] // Column
Out[ ]= {{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}}
```

The cross product:

```
In[ ]:= LeviCivitaTensor[3].{x1, x2, x3}.{y1, y2, y3}
Out[ ]= {x3 y2 - x2 y3, -x3 y1 + x1 y3, x2 y1 - x1 y2}
```

Same as

```
In[ ]:= Cross[{x1, x2, x3}, {y1, y2, y3}]
Out[ ]= {-x3 y2 + x2 y3, x3 y1 - x1 y3, -x2 y1 + x1 y2}
```

Compute $F_{ij=}\,\epsilon_{ijk}\,B_k$, the spatial part of the EM tensor $F_{\mu\nu}$:

First construct the tensor $\epsilon \otimes \omega$ (⊗ is [Esc]t*[Esc], *different* from [Esc]c*[Esc] → ⊗)

In[ ]:=
```
B = {B1, B2, B3};
LeviCivitaTensor[3, List] ⊗ B
```

Out[ ]=
```
{{{{0, 0, 0}, {0, 0, 0}, {0, 0, 0}},
  {{0, 0, 0}, {0, 0, 0}, {B1, B2, B3}}, {{0, 0, 0}, {-B1, -B2, -B3}, {0, 0, 0}}},
 {{{0, 0, 0}, {0, 0, 0}, {-B1, -B2, -B3}}, {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}},
  {{B1, B2, B3}, {0, 0, 0}, {0, 0, 0}}}, {{{0, 0, 0}, {B1, B2, B3}, {0, 0, 0}},
  {{-B1, -B2, -B3}, {0, 0, 0}, {0, 0, 0}}, {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}}}}
```

Same as using the function TensorProduct:

In[ ]:=
```
TensorProduct[LeviCivitaTensor[3, List], B]
```

Out[ ]=
```
{{{{0, 0, 0}, {0, 0, 0}, {0, 0, 0}},
  {{0, 0, 0}, {0, 0, 0}, {B1, B2, B3}}, {{0, 0, 0}, {-B1, -B2, -B3}, {0, 0, 0}}},
 {{{0, 0, 0}, {0, 0, 0}, {-B1, -B2, -B3}}, {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}},
  {{B1, B2, B3}, {0, 0, 0}, {0, 0, 0}}}, {{{0, 0, 0}, {B1, B2, B3}, {0, 0, 0}},
  {{-B1, -B2, -B3}, {0, 0, 0}, {0, 0, 0}}, {{0, 0, 0}, {0, 0, 0}, {0, 0, 0}}}}
```

Now we can use the function TensorContract to obtain $\epsilon_{ijk} B_k$

In[ ]:=
```
TensorContract[LeviCivitaTensor[3, List] ⊗ B, {3, 4}] // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} 0 & B3 & -B2 \\ -B3 & 0 & B1 \\ B2 & -B1 & 0 \end{pmatrix}$$

Compute the magnetic field $B = \nabla \times A$ or $B_i = \epsilon_{ijk} \partial_j A_k$ ([Esc]del[Esc]→∇, [Esc]pd[Esc]→∂, [Esc]*[Esc]→ ×)

Now we contract two pairs of indices:

In[ ]:=
```
ϵ = LeviCivitaTensor[3, List];
δ = {"∂"1, "∂"2, "∂"3}; (* use ∂ as character, not as partial derivative*)
a = {A1, A2, A3};
t1 = TensorProduct[ϵ, δ, a];
TensorContract[t1, {{2, 4}, {3, 5}}] // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -\partial_3 A_2 + \partial_2 A_3 \\ \partial_3 A_1 - \partial_1 A_3 \\ -\partial_2 A_1 + \partial_1 A_2 \end{pmatrix}$$

Same as:

In[ ]:=
```
Cross[δ, a] // MatrixForm
```

Out[ ]//MatrixForm=

$$\begin{pmatrix} -\partial_3 A_2 + \partial_2 A_3 \\ \partial_3 A_1 - \partial_1 A_3 \\ -\partial_2 A_1 + \partial_1 A_2 \end{pmatrix}$$

Prove identity $\epsilon_{ijk}\,\epsilon_{lmk} = \delta_{il}\,\delta_{jm} - \delta_{im}\,\delta_{jl}$

In[ ]:=
```
ϵ   = LeviCivitaTensor[3, List];
ϵϵ  = ϵ⊗ϵ;
ϵ2  = TensorContract[ϵϵ, {3, 6}];
lhs = Table[ϵ2[[i, j, l, m]] - (KroneckerDelta[i, l] KroneckerDelta[j, m] -
       KroneckerDelta[i, m] KroneckerDelta[j, l]), {i, 3}, {j, 3}, {l, 3}, {m, 3}];
rhs = ConstantArray[0, {3, 3, 3, 3}]; (*a nested list filled with zeroes*)
lhs == rhs
```

Out[ ]=
```
True
```