Pauli Matrices

Mauro Mongiardo[1]

[1] Department of Engineering, University of Perugia, Perugia, Italy.

Table of contents

Introduction

Traditional vector analysis relies on the approach proposed by Gibbs at the beginning of 1900 [1] and generally adopted in electromagnetic field engineering.

Later, several different approaches have been presented and elucidated.

As an example differential forms [2] and Geometric or Clifford Algebra, in the following briefly referred to as GA [3][4].

Generally GA has been employed in electromagnetic (EM) by considering the point of view of physicists and not of engineers [5][6].

As an example, in EM engineering wide use is made of time–harmonic analysis [8], while no publication deals with application of GA to the time–harmonic regime.

References I

[1] J. W. Gibbs, *Elements of Vector Analysis*, Tuttle, Morehouse & Taylor, 1884;

[2] P. Russer, *Exterior Differential Forms in Teaching Electromagnetics* in *Electromagnetics in a Complex World - Challenges and Perspectives*, Springer, 2004;

[3] D. Hestenes, G. Sobczyk, *Clifford Algebra to Geometric Calculus: A Unified Language for Mathematics and Physics*, Dordrecht: Kluwer Academic Publishers, 1987;

[4] A. Seagar, *Application of Geometric Algebra to Electromagnetic scattering: The Clifford–Cauchy–Dirac Technique*, Springer Publishing Companys, 2015;

[5] J. W. Arthur, *Understanding Geometric Algebra*, New York: Wiley-IEEE, 2011;

References II

[6] J. M. Chappell et al., "Geometric Algebra for Electrical and Electronic Engineers," in Proceedings of the IEEE, vol. 102, no. 9, pp. 1340-1363, Sept. 2014. doi: 10.1109/JPROC.2014.2339299;

[7] J. M. Chappell, A. Iqbal, J. G. Hartnett and D. Abbott, "The Vector Algebra War: A Historical Perspective," in IEEE Access, vol. 4, no. , pp. 1997-2004, 2016. doi: 10.1109/ACCESS.2016.2538262

[8] R.F. Harrington, *Time-Harmonic Electromagnetic Fields*, IEEE-Press, 2001;

As appeared in [7]

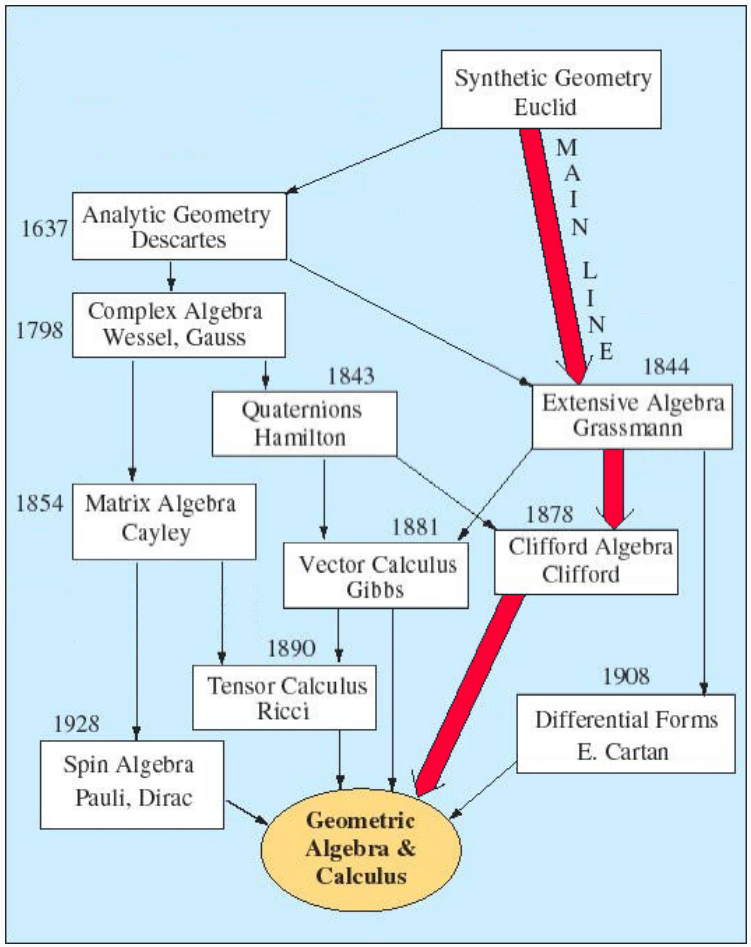

Figure 1: The descent of the various vector systems. The main path of development beginning from Euclid geometry down through Grassman and then to Clifford. Other parallel developments using complex numbers, quaternions, Gibbs' vectors, tensors, matrices and spinor algebra subsumed into the general formalism of Clifford geometric algebra with the inclusion of calculus.

Most of the engineering EM analysis are performed in a three–dimensional space (3D). In the 3D space a very interesting fact takes place: GA is equivalent to Pauli algebra.

Pauli matrices have been introduced by Wolfgang Pauli for the spin theory.

Very remarkably, by using Pauli matrices a vector can be represented as a 2x2 matrix. Thus it is possible to multiply vectors by multiplying the relative matrices.

It is also possible to make the inverse of a vector!

GA in 3D and Pauli matrices

In the 3D space, according to GA, we need to represent:

- a scalar;
- 3 vectors;
- 3 bivectors;
- psuedo–scalar

i.e. 8 numbers.

It is possible to introduce a multivector which is the sum of a scalar, a vector, a bivector and a pseudo–scalar.

Such multivector can be represented by a 2x2 matrix and can be constructed in terms of Pauli matrices.

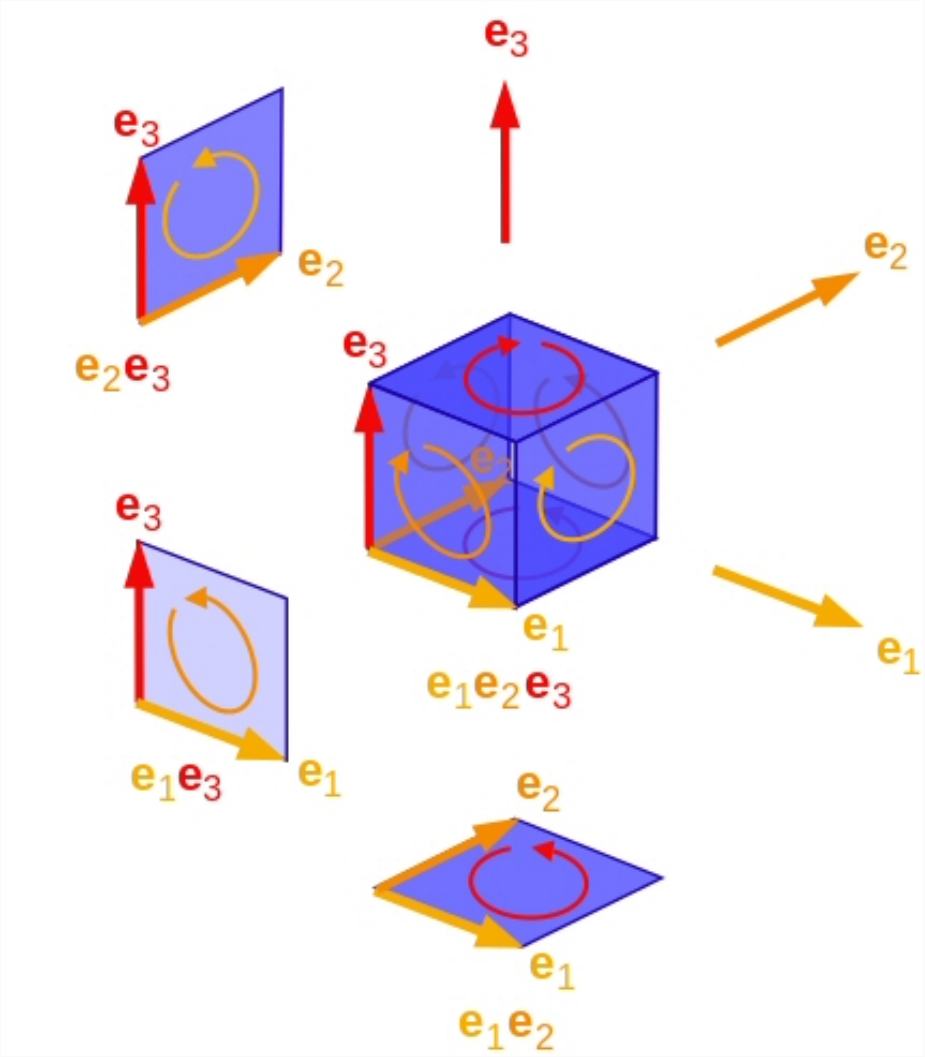

Figure 2: Identify the basis vectors as $e_1 = \sigma_1$, $e_2 = \sigma_2$, $e_3 = \sigma_3$.These are elements of Clifford's model for three–dimensional space. This consists of three unit vectors e_1, e_2 and e_3, three unit areas e_2e_3, e_3e_1 and e_1e_2 and a unit volume $i = e_1e_2e_3$. The pure scalars then defining points to form a complete algebraic description of three-dimensional physical space.

Pauli matrices and their properties

Wolfgang Pauli

Figure 3: Wolfgang Ernst Pauli (25 April 1900 – 15 December 1958)

Wolfgang Pauli was an Austrian-born Swiss and American theoretical physicist and one of the pioneers of quantum physics.

In 1945, after having been nominated by Albert Einstein, Pauli received the Nobel Prize in Physics for his "decisive contribution through his discovery of a new law of Nature, the exclusion principle or Pauli principle".

Pauli matrices

We first introduce the Pauli matrices and then discuss some of their properties and show how to operate with this new tool.

Note that engineers are quite well trained to operate with matrices and we will see that many standard vectors operations can be simplified and new important elements will be found.

The Pauli matrices have the following form

$$\begin{aligned} \sigma_1 &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \\ \sigma_2 &= \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \\ \sigma_3 &= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} . \end{aligned} \tag{1}$$

Pauli matrices

The Pauli matrices are a set of three 2x2 complex matrices which are *Hermitian* and *unitary*.

An Hermitian matrix (or *self-adjoint* matrix) is a complex square matrix that is equal to its own conjugate transpose, that is, the element in the i-th row and j-th column is equal to the complex conjugate of the element in the j-th row and i-th column, for all indices i and j. Let us consider the matrix a

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

its transpose is given by

$$A^T = \begin{pmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{pmatrix}$$

By denoting with * the complex conjugate, its Hermitian $A^\dagger$ is

$$A^\dagger = \begin{pmatrix} a_{11}^* & a_{21}^* \\ a_{12}^* & a_{22}^* \end{pmatrix}$$

The *trace* of A is

$$tr(A) = a_{11} + a_{22}\,. \tag{2}$$

A complex square matrix U is unitary if its conjugate transpose $U^\dagger$ is also its inverse.

Properties of the Pauli matrices

It is immediately noted that the trace of these matrices is always zero.

The determinant of the Pauli matrices is always -1.

The following products hold:

$$\sigma_1^2 = \sigma_2^2 = \sigma_3^2 = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I = \sigma_0 \,. \tag{3}$$

By multiplying e.g. σ_1 with σ_2 the result is $i\sigma_3$ and similarly for the other cases:

$$\begin{aligned} \sigma_1\sigma_2 &= i\sigma_3 = -\sigma_2\sigma_1 \\ \sigma_2\sigma_3 &= i\sigma_1 = -\sigma_3\sigma_2 \\ \sigma_3\sigma_1 &= i\sigma_2 = -\sigma_1\sigma_3 \,. \end{aligned} \tag{4}$$

The above relations are very important. In fact, they show that, in the three-dimensional case, we can always replace the quantities $\sigma_i\sigma_j$ with the orthogonal vector $i\sigma_k$.

From the above properties it is seen that

$$(\sigma_1\sigma_2\sigma_3)^2 = \sigma_1\sigma_2\sigma_3\sigma_1\sigma_2\sigma_3 = -1 \tag{5}$$

i.e. $\sigma_1\sigma_2\sigma_3 = i$.

The three Pauli matrices, with the addition of the identity matrix σ_0, form a basis in the space of the 2x2 Hermitian matrices and a matrix A can be represented as:

$$A = a_0\sigma_0 + a_1\sigma_1 + a_2\sigma_2 + a_3\sigma_3\,. \tag{6}$$

It is worthwhile to note that when the coefficients (a_0, a_1, a_2, a_3) are complex, also non Hermitian matrices can be described by the basis of $(\sigma_0, \sigma_1, \sigma_2, \sigma_3)$.

The Pauli vector

The Pauli vector

Let us introduce the Pauli vector which is a vector made by the three matrices $(\sigma_1, \sigma_2, \sigma_3)$.

$$\sigma = (\sigma_1 \, \mathbf{x}_0, \sigma_2 \, \mathbf{y}_0, \sigma_3 \, \mathbf{z}_0) \; . \tag{7}$$

We can consider the vector $\mathbf{a} = (a_x \, \mathbf{x}_0 + a_y \, \mathbf{y}_0 + a_z \, \mathbf{z}_0)$ in the three-dimensional space and make the following product:

$$\begin{aligned}
\tilde{a} &= \sigma \cdot \mathbf{a} = \sigma_1 a_x + \sigma_2 a_y + \sigma_3 a_z && (8) \\
&= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} a_x + \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} a_y + \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} a_z && (9) \\
&= \begin{pmatrix} a_z & a_x - i a_y \\ a_x + i a_y & -a_z \end{pmatrix} && (10)
\end{aligned}$$

The matrix $\tilde{a}$ is an equivalent description of the vector $\mathbf{a}$ in terms of a 2x2 matrix. Two different symbols have been used to denote the 2x2 matrix representation $\tilde{a}$ and the standard vector representation $\mathbf{a}$.

Pauli matrices properties determination with a computer algebra system

It is instructive to try to verify the properties previously illustrated by using a computer algebra system (CAS). Here following we use wxMaxima to define the 2x2 Pauli matrices and to perform some computations. Please refer to the file:

`wxMaxima/Pauli_def.wxm`

which is listed in the following.

Pauli definitions

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$
/* Pauli_def */

print("------------------------------------------------------------")$
print("Pauli Matrices Definition")$
print("------------------------------------------------------------")$
%sigma[0] : matrix([1,0],[0,1])$
%sigma[1] : matrix([0,1],[1,0])$
%sigma[2] : matrix([0,-%i],[%i,0])$
%sigma[3] : matrix([1,0],[0,-1])$
print("%sigma[0]," = ",%sigma[0])$
print("%sigma[1]," = ",%sigma[1])$
print("%sigma[2]," = ",%sigma[2])$
print("%sigma[3]," = ",%sigma[3])$


print("------------------------------------------------------------")$
print("Pauli Matrices Properties")$
print("------------------------------------------------------------")$
print("")$
print("------------------------------------------------------------")$
print("Squared give the identity matrix %sigma[i].%sigma[i]=sigma[0]")$
print("------------------------------------------------------------")$

print("involutory (it is also its inverse)")$
ss1 : %sigma[1] . %sigma[1];
ss2 : %sigma[2] . %sigma[2];
ss3 : %sigma[3] . %sigma[3];

print("------------------------------------------------------------")$
print(" %sigma[1] . %sigma[2] =  i %sigma[3]")$
print("------------------------------------------------------------")$
print("Let us try multiply %sigma[1].%sigma[2]")$
%sigma[1] . %sigma[2];
print("It is apparent that the result is i %sigma[3]")$
print("------------------------------------------------------------")$
print("Let us try multiply %sigma[1].%sigma[3]")$
%sigma[1] . %sigma[3];
print("It is apparent that the result is - i %sigma[2]")$
print("------------------------------------------------------------")$
print("Let us try multiply %sigma[2].%sigma[3]")$
%sigma[2] . %sigma[3];
print("It is apparent that the result is i %sigma[1]")$


print("------------------------------------------------------------")$
print("Anticommutative")$
print("------------------------------------------------------------")$
print("Commutation : %sigma[1] . %sigma[2] + %sigma[2] . %sigma[1]")$
sc12 : %sigma[1] . %sigma[2] + %sigma[2] . %sigma[1];
print("The result is 0 %sigma[0]")$
print("------------------------------------------------------------")$
print("If we add instead of subtracting")$
print(" %sigma[1] . %sigma[2] + %sigma[2] . %sigma[1]")$
sc12 : %sigma[1] . %sigma[2] + %sigma[2] . %sigma[1];
print("the result is %sigma[1] . %sigma[2] = - %sigma[2] . %sigma[1]")$
/*
print("------------------------------------------------------------")$
print("(%sigma[i] . %sigma[j]) squared gives -1 * %sigma[0]")$
print("------------------------------------------------------------")$
print("Squared %sigma[1] . %sigma[2]")$
sc12 : (%sigma[1] . %sigma[2]) . (%sigma[1] . %sigma[2]);
print("The result is -1 * %sigma[0] !!")$

print("------------------------------------------------------------")$
print("%sigma[1] . %sigma[2] . %sigma[3] = i * %sigma[0]")$
print("------------------------------------------------------------")$
print("Squared %sigma[1] . %sigma[2] . %sigma[3]")$
sc12 : (%sigma[1].%sigma[2].%sigma[3]).(%sigma[1].%sigma[2].%sigma[3]);
print("The result is -%sigma[0] !!")$

*/

print("end")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```

Product

We introduce the Pauli matrices and then the representation of two vectors **a** and **b** via their Pauli matrices $\tilde{a}, \tilde{b}$. Then we perform their product obtaining $\tilde{c} = \tilde{a}\tilde{b}$ as illustrated in the code :

`wxMaxima/Pauli_ab.wxm`

which is listed in the following.

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$

print("------------------------------------------------------------")$
print("Product of two vectors via Pauli Matrices")$
print("------------------------------------------------------------")$

declare([a1,a2,a3],scalar)$
declare([b1,b2,b3],scalar)$


%sigma[0] : matrix([1,0],[0,1])$
%sigma[1] : matrix([0,1],[1,0])$
%sigma[2] : matrix([0,-%i],[%i,0])$
%sigma[3] : matrix([1,0],[0,-1])$

print("------------------------------------------------------------")$
print("Pauli Matrices Definition")$
print("------------------------------------------------------------")$

print('%sigma[0],"  = ",%sigma[0])$
print('%sigma[1],"  = ",%sigma[1])$
print('%sigma[2],"  = ",%sigma[2])$
print('%sigma[3],"  = ",%sigma[3])$

a : a1 . %sigma[1] + a2 . %sigma[2] + a3 . %sigma[3]$
print("Express vector ",'a," as a Pauli matrix ",a)$

b : b1 . %sigma[1] + b2 . %sigma[2] + b3 . %sigma[3]$
print("Express vector ",'b," as a Pauli matrix ",b)$

print("Find their product")$
c : ratsimp(a . b);


print("The compoments of ab are found as follows")$
print("The  scalar part (inner product) is the trace/2")$
sc : ratsimp((c[1,1]+c[2,2])/2);


print("------------------------------------------------------------")$
print("The above is the dot product a . b  ")$
print("and it is the trace/2 of the matrix ")$
print("------------------------------------------------------------")$

print("The  x component is given by ",('c[21]+'c[12])/2)$
cx : factor(ratsimp((c[2,1]+c[1,2])/2));

print("The  y component is given by ", ('c[21]-'c[12])/(2*%i))$
cy : factor(ratsimp((c[2,1]-c[1,2])/(2*%i)));

print("The  z component is given by ", ('c[11]-'c[22])/2)$
cz : factor(ratsimp((c[1,1]-c[2,2])/2));

print("one can recognizes that the above components,")$
print("apart for an i factor, are the same of a x b ")$


print("This new part is called external product")$
print("The external product ",'a ,"^", 'b)$
awb : factor(ratsimp(c - sc.%sigma[0]));


print("------------------------------------------------------------")$
print("The external product a ^ b = i a x b")$
print("------------------------------------------------------------")$

print("It is also possible to compute the determinant of a vector")$
deta : ratsimp(determinant(a));


print("its modulus is the square root of the abs value of the determinant")$
sqrt(abs(deta));

print("and its inverse is")$
inva : ratsimp(invert(a));
print("note that the direction is the same of the original vector")$
print("but it is scaled by the absolute value of the determinant")$

print("end")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```

Let us try the product between two matrices $\tilde{a}, \tilde{b}$ and evaluate their product $\tilde{c} = \tilde{a}\tilde{b}$,

$$\begin{aligned}
\tilde{a} &= \begin{pmatrix} a_3 & a_1 - i\,a_2 \\ i\,a_2 + a_1 & -a_3 \end{pmatrix} \\
\tilde{b} &= \begin{pmatrix} b_3 & b_1 - i\,b_2 \\ i\,b_2 + b_1 & -b_3 \end{pmatrix} \\
\tilde{c} &= \begin{pmatrix} a_3\,b_3 + (a_2 + i\,a_1)\,b_2 + (a_1 - i\,a_2)\,b_1 & (i\,a_2 - a_1)\,b_3 - i\,a_3\,b_2 + a_3\,b_1 \\ (i\,a_2 + a_1)\,b_3 - i\,a_3\,b_2 - a_3\,b_1 & a_3\,b_3 + (a_2 - i\,a_1)\,b_2 + (i\,a_2 + a_1)\,b_1 \end{pmatrix}
\end{aligned}$$

- *the trace of $\tilde{c} = \tilde{a}\tilde{b}$ divided by 2 gives us the dot product*.

$$\mathbf{a} \cdot \mathbf{b} = (\tilde{c}_{11} + \tilde{c}_{22})/2 = a_1\,b_1 + a_2\,b_2 + a_3\,b_3 \tag{11}$$

It is straightforward to recognize that the component along x, y, z can be easily retrieved by the following operations

$$\begin{aligned}
c_x &= \frac{c_{21} + c_{12}}{2} \\
c_y &= -\frac{i\,(c_{21} - c_{12})}{2} \\
c_z &= \frac{c_{11} - c_{22}}{2}
\end{aligned} \tag{12}$$

By writing them explicitly we find:

$$\begin{aligned} c_x &= i\,(a_2\,b_3 - a_3\,b_2) \\ c_y &= -i\,(a_1\,b_3 - a_3\,b_1) \\ c_z &= i\,(a_1\,b_2 - a_2\,b_1) \end{aligned} \tag{13}$$

It is now easy to recognize that, apart for the i factor, this is equal to the well known cross product. This new part is called *external product* and is denoted by $\mathbf{a} \wedge \mathbf{b}$. We have just obtained the important identity:

$$\mathbf{a} \wedge \mathbf{b} = i\,\mathbf{a} \times \mathbf{b}\,. \tag{14}$$

Naturally, to represent the wedge product in terms of the Pauli matrices it is sufficient to subtract the dot product from $\tilde{a}\tilde{b}$, thus obtaining

$$\mathbf{a} \wedge \mathbf{b} = \begin{pmatrix} i\,(a_1\,b_2 - a_2\,b_1) & i\,a_2\,b_3 - a_1\,b_3 - i\,a_3\,b_2 + a_3\,b_1 \\ i\,a_2\,b_3 + a_1\,b_3 - i\,a_3\,b_2 - a_3\,b_1 & -i\,(a_1\,b_2 - a_2\,b_1) \end{pmatrix}. \tag{15}$$

The quantity $\mathbf{a} \wedge \mathbf{b}$ is a *bivector* .

The product $\tilde{b}\tilde{a}$

It is immediate, by applying the rules of matrix multiplication, to compute the product $\tilde{b}\tilde{a}$. An example of code doing this is reported in:

`wxMaxima/Pauli_ba.wxm`

which is listed in the following.

$\tilde{b}, \tilde{a}$

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$

/* Pauli_ba */

declare([a1,a2,a3],scalar)$
declare([b1,b2,b3],scalar)$

%sigma[0] : matrix([1,0],[0,1])$
%sigma[1] : matrix([0,1],[1,0])$
%sigma[2] : matrix([0,-%i],[%i,0])$
%sigma[3] : matrix([1,0],[0,-1])$

a : a1 . %sigma[1] + a2 . %sigma[2] + a3 . %sigma[3]$
print("Express vector ",'a," as a Pauli matrix ",a)$
b : b1 . %sigma[1] + b2 . %sigma[2] + b3 . %sigma[3]$
print("Express vector ",'b," as a Pauli matrix ",b)$

print("Find the product ba")$
ba : ratsimp(b . a);

print("The compoments of ba are found as follows")$
print("The  scalar part (inner product) is the trace/2")$
sba : ratsimp((ba[1,1]+ba[2,2])/2);

print("------------------------------------------------------------")$
print("The above is the dot product b . a = a . b")$
print("      and it is the trace/2 of the matrix ")$
print("------------------------------------------------------------")$

print("The  x component is given by ",('ba[2,1]+'ba[1,2])/2)$
bax : factor(ratsimp((ba[2,1]+ba[1,2])/2));

print("The  y component is given by ", ('ba[2,1]-'ba[1,2])/(2*%i))$
bay : factor(ratsimp((ba[2,1]-ba[1,2])/(2*%i)));

print("The  z component is given by ", ('ab[1,1]-'ab[2,2])/2)$
baz : factor(ratsimp((ba[1,1]-ba[2,2])/2));

print("the above components are the same of - i(a x b)")$

print("This new part is called external product")$
print("The external product ",'b ,"^", 'a)$
bwa : factor(ratsimp(ba - sba.%sigma[0]));

print("------------------------------------------------------------")$
print("Note that the external product b ^ a = - a ^ b")$
print("The external product is anticommutative")$
print("------------------------------------------------------------")$

print("------------------------------------------------------------")$
print("Therefore the dot product a . b can be obtained as (ab + ba)/2")$
print("While the external product a ^ b can be obtained as (ab - ba)/2")$
print("------------------------------------------------------------")$

print("end")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```

$\tilde{b}, \tilde{a}$

Naturally, the Pauli matrices and the matrices $\tilde{a}, \tilde{b}$ are the same as before. However, now we have:

$$\tilde{d} \quad = \quad \tilde{b}\tilde{a} = \begin{pmatrix} a_3\, b_3 + (a_2 - i\, a_1)\; b_2 + (i\, a_2 + a_1)\; b_1 & (a_1 - i\, a_2)\; b_3 + i\, a_3\, b_2 - a_3\, b_1 \\ (-i\, a_2 - a_1)\; b_3 + i\, a_3\, b_2 + a_3\, b_1 & a_3\, b_3 + (a_2 + i\, a_1)\; b_2 + (a_1 - i\, a_2)\; b_1 \end{pmatrix}$$

It is apparent that the trace is the same as the one from $\tilde{a}\tilde{b}$.

By writing explicitly the various components we find:

$$\begin{aligned} d_x &= -i\,(a_2\, b_3 - a_3\, b_2) \\ d_y &= i\,(a_1\, b_3 - a_3\, b_1) \\ d_z &= -i\,(a_1\, b_2 - a_2\, b_1) \end{aligned} \tag{16}$$

Relation between cross and external product

It is now easy to recognize that these are the coefficient of $-i\,\mathbf{a}\times\mathbf{b}$. The *external product* $\mathbf{b}\wedge\mathbf{a}$ thus satisfy the rule:

$$\mathbf{b}\wedge\mathbf{a} = -\mathbf{a}\wedge\mathbf{b} = -i\,\mathbf{a}\times\mathbf{b}\,. \tag{17}$$

We have another relevant property: *the external product between two vectors is anti–commutative*.

Alternative definitions

We have therefore found that the dot product can also be obtained as:

$$\mathbf{a} \cdot \mathbf{b} = \frac{\tilde{a}\tilde{b} + \tilde{b}\tilde{a}}{2}\,, \tag{18}$$

while for the external product we have:

$$\mathbf{a} \wedge \mathbf{b} = \frac{\tilde{a}\tilde{b} - \tilde{b}\tilde{a}}{2}\,. \tag{19}$$

The relations (**??**), (**??**) can be also taken as the definitions of dot and external products, respectively.

Inner product

The inner product between two matrices A and B can be defined as following

$$\langle A, B \rangle = \frac{1}{2} tr\left(AB^{\dagger}\right). \tag{20}$$

From the properties of the Pauli matrices it is readily recognized that, for $i \neq j$ we have $\langle \sigma_i, \sigma_j \rangle = 0$, while $\langle \sigma_i, \sigma_i \rangle = 1$. This provides a simple way to retrieve the coefficients of the Pauli matrices for a given matrix A. In fact, we have

$$\langle A, \sigma_0 \rangle \quad = \quad a_0 \langle \sigma_0, \sigma_0 \rangle + a_1 \langle \sigma_1, \sigma_0 \rangle + a_2 \langle \sigma_2, \sigma_0 \rangle + a_3 \langle \sigma_3, \sigma_0 \rangle = a_0$$

and similarly for other components.

Once we have represented the vector $\mathbf{a}$ as a matrix $\tilde{a}$ it is possible to compute its determinant and its inverse.

It is immediately recognized that we have for the determinant

$$det(\tilde{a}) = -(a_1^2 + a_2^2 + a_3^2) \tag{21}$$

from which it is evident that, by taking the square root of the absolute value, we can recover the modulus of the vector.

In standard vector algebra the inverse of a vector is not defined. However we can perform the inverse of $\tilde{a}$ obtaining

$$\tilde{a}^{-1} = \frac{1}{{a_1}^2 + {a_2}^2 + {a_3}^2} \begin{pmatrix} a_3 & -i\,a_2 + a_1 \\ i\,a_2 + a_1 & -a_3 \end{pmatrix} \tag{22}$$

This is just the same vector divided by the square of its modulus!

Finding the operator which transforms one vector into another

Let us assume that the vectors $\mathbf{a}, \mathbf{b}$ are given and we want to find the operator that transforms vector $\mathbf{a}$ into vector $\mathbf{b}$.

In conventional vector analysis, apart for particular cases, this is not a simple operation.

Using Pauli matrices it is quite trivial. In fact, by calling with $\tilde{c}$ the transformation we have that

$$\tilde{b} = \tilde{c}\tilde{a} \tag{23}$$

The transformation is simply

$$\tilde{c} = \tilde{b}\tilde{a}^{-1}\,. \tag{24}$$

Triple products with Pauli matrices

Let us now investigate the triple product.

We start by considering the product $\mathbf{a}(\mathbf{b} \wedge \mathbf{c})$ and by expressing $\mathbf{b} \wedge \mathbf{c}$.

In (**??**) the geometric product has been written for $\mathbf{a} \wedge \mathbf{b}$ and it is now repeated giving:

$$\mathbf{b}\wedge\mathbf{c} = \begin{pmatrix} i\,b_1\,c_2 - i\,b_2\,c_1 & (i\,b_2 - b_1)\,c_3 - i\,b_3\,c_2 + b_3\,c_1 \\ (i\,b_2 + b_1)\,c_3 - i\,b_3\,c_2 - b_3\,c_1 & i\,b_2\,c_1 - i\,b_1\,c_2 \end{pmatrix}. \tag{25}$$

The next step is to obtain $d = \mathbf{a}(\mathbf{b} \wedge \mathbf{c})$ which simply reduces to the matrix multiplication of $\mathbf{a}$ and $\mathbf{b} \wedge \mathbf{c}$.

This operation, together with a few others is reported in the following code.

Note that we have used the symbol d to denote the result but we still need to identify what type of result we are going to get.

It will be a 2x2 matrix but the meaning of its components is still to be found. Please refer to the following code as an example:

`wxMaxima/Pauli_abc.wxm`

triple product

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$
load(vect);

declare([a1,a2,a3],scalar)$
declare([b1,b2,b3],scalar)$
declare([c1,c2,c3],scalar)$

print("  ")$
print("Pauli Matrices Definition")$
print("  ")$

%sigma[0] : matrix([1,0],[0,1])$
%sigma[1] : matrix([0,1],[1,0])$
%sigma[2] : matrix([0,-%i],[%i,0])$
%sigma[3] : matrix([1,0],[0,-1])$

print("%sigma[0] =","%sigma[0]")$
print("%sigma[1] =","%sigma[1]")$
print("%sigma[2] =","%sigma[2]")$
print("%sigma[3] =","%sigma[3]")$

a : a1 . %sigma[1] + a2 . %sigma[2] + a3 . %sigma[3]$
print("Express vector ","a","as a Pauli matrix ",a)$

b : b1 . %sigma[1] + b2 . %sigma[2] + b3 . %sigma[3]$
print("Express vector ","b","as a Pauli matrix ",b)$

c : c1 . %sigma[1] + c2 . %sigma[2] + c3 . %sigma[3]$
print("Express vector ","c","as a Pauli matrix ",c)$

print("  ")$
print("geometric (Clifford) product bc (between two vectors)")$
print("  ")$

print("Find their product")$
bc : ratsimp(b . c);

print("The  scalar part (inner product) is the trace/2")$
sbc : ratsimp((bc[1,1]+bc[2,2])/2);

print("The external product ","b ^^ ","c")$
bwc : ratsimp(bc - sbc *%sigma[0]);
print("one recognizes that the above is %i b x c")$
print("  ")$
print("  The following identity holds b ^ c = %i b x c  ")$
print("  ")$
print("Now perform a (b ^ c)")$
print("  ")$
abwc : a . bwc;

print("The  scalar part (inner product) is the trace/2")$
sbc : factor(ratsimp((abwc[1,1]+abwc[2,2])/2));
print("Note that the above is a trivector (a pseudoscalar)")$
print("It corresponds to the volume element a ^ b ^ c")$

print("If we consider the matrix ")$
abc : matrix([a1,a2,a3],[b1,b2,b3],[c1,c2,c3])$
print("abc," = ",abc)$
print("its determinat is ")$
dabc : determinant(abc);
print("The determinant multiplied by i is equal to the scalar part.")$
print("their difference being")$
ratsimp(%i * dabc - sbc);

print("  ")$
print("Therefore we have a ^ b ^ c = i det(abc)")$
print("  ")$

print("The remaining part corresponds to a dot (b ^ c)")$
awbwc : ratsimp(abwc - sbc . %sigma[0]);

print("i.e. a vector with components")$
wx : ratsimp((awbwc[1,1]+awbwc[2,2])/2)$
wx : ratsimp((awbwc[2,1]+awbwc[1,2])/2)$
wy : ratsimp((awbwc[2,1]-awbwc[1,2])/(2*%i))$
wz : ratsimp((awbwc[1,1]-awbwc[2,2])/2)$

print("wx, " = ",wx )$
print("wx, " = ",wx )$
print("wy, " = ",wy )$
print("wz, " = ",wz )$

print("  ")$
print(" (a.b)c - (a.c)b  (identity bac- cab = a x b c )  ")$
print("  ")$

print("trying a different approach")$
print("evaluating dot products e.g. a dot b as (ab + ba)/2")$
adb : ratsimp((a . b + b . a)/2);
adc : ratsimp((a . c + c . a)/2);
w : adb . c - adc . b;

ax : ratsimp((w[1,1]+w[2,2])/2)$
ax : ratsimp((w[2,1]+w[1,2])/2)$
ay : ratsimp((w[2,1]-w[1,2])/(2*%i))$
az : ratsimp((w[1,1]-w[2,2])/2)$

print("i.e. a vector with components")$
print("ax, " = ",ax )$
print("ax, " = ",ax )$
print("ay, " = ",ay )$
print("az, " = ",az )$

print("Proof that are equal")$
ratsimp(wx-ax);
ratsimp(wy-ay);
ratsimp(wz-az);

print("  ")$
print("Therefore w = a . (b ^ c) = (a.b)c - (a.c)b")$
print("  ")$

print("=========================GIBBS=========================")$
a :[a1, a2, a3];
b :[b1, b2, b3];
c : [c1, c2, c3];
/*
print(" dot product")$
c : a . b;

print(" cross product")$
 axb : a ^ b;
 axb : express(axb);

bxa : express(b ^ a);
print("find the difference axb - bxa")$
diff : axb + bxa;
*/

print("A X ( B X C ) = ")$
r1 : express(a ^ express(b ^ c))$
r1 : ratsimp(r1);

print("The sum of these coefficients with")$
print("those previously found gives zero ")$
ratsimp(r1[1] + ax);
ratsimp(r1[2] + ay);
ratsimp(r1[3] + az);

print("  ")$
print("Therefore a . (b ^ c) = (a.b)c - (a.c)b == - a x b x c  ")$
print("  ")$

print("end")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```

Scalar part

After performing the matrix multiplication one get

$$\begin{aligned} d &= \mathbf{a}(\mathbf{b}\wedge\mathbf{c}) \\ d_{11} &= (a_1 - i\,a_2)\,((i\,b_2 + b_1)\,c_3 - i\,b_3\,c_2 - b_3\,c_1) + a_3\,(i\,b_1\,c_2 - i\,b_2\,c_1) \\ d_{12} &= a_3\,((i\,b_2 - b_1)\,c_3 - i\,b_3\,c_2 + b_3\,c_1) + (a_1 - i\,a_2)\,(i\,b_2\,c_1 - i\,b_1\,c_2) \\ d_{21} &= (i\,a_2 + a_1)\,(i\,b_1\,c_2 - i\,b_2\,c_1) - a_3\,((i\,b_2 + b_1)\,c_3 - i\,b_3\,c_2 - b_3\,c_1) \\ d_{22} &= (i\,a_2 + a_1)\,((i\,b_2 - b_1)\,c_3 - i\,b_3\,c_2 + b_3\,c_1) - a_3\,(i\,b_2\,c_1 - i\,b_1\,c_2) \end{aligned} \tag{26}$$

As before the scalar part of d is obtained by taking half of the trace and is denoted here (for reasons that will become clear in the following) as $\langle d\rangle_3$

$$\langle d\rangle_3 = i\,(a_1\,b_2\,c_3 - a_2\,b_1\,c_3 - a_1\,b_3\,c_2 + a_3\,b_1\,c_2 + a_2\,b_3\,c_1 - a_3\,b_2\,c_1) \tag{27}$$

It is noted that there is the factor i in front of the expression.

This part corresponds to a *trivector*, i.e. to the volume element obtained by performing $\mathbf{a}\wedge\mathbf{b}\wedge\mathbf{c}$.

pseudoscalar

It is also observed that if we consider the matrix

$$abc = \begin{pmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{pmatrix} \tag{28}$$

its determinant is

$$det(abc) = a_1\ (b_2\, c_3 - b_3\, c_2) - a_2\ (b_1\, c_3 - b_3\, c_1) + a_3\ (b_1\, c_2 - b_2\, c_1) \tag{29}$$

and, when multiplied by i coincides with (**??**). We have therefore derived the important property:

$$\mathbf{a} \wedge \mathbf{b} \wedge \mathbf{c} = i\, det\,(abc) = \langle d \rangle_3 \tag{30}$$

It is noted that $\mathbf{a} \wedge \mathbf{b} \wedge \mathbf{c}$ is often called *pseudoscalar* i.e. is a scalar quantity multiplied by i. In the 3D space this is the element of the third grade (this is the reason why we have used $\langle d \rangle_3$). This is also the highest grade in 3D.

observations

A few observations are in order:

$$(\mathbf{a} \wedge \mathbf{b}) \wedge \mathbf{c} = \mathbf{a} \wedge (\mathbf{b} \wedge \mathbf{c}) \tag{31}$$

$$\mathbf{a} \wedge \mathbf{b} \wedge \mathbf{c} \wedge \mathbf{d} = 0 \tag{32}$$

In (**??**) it is noted that, *unlikely to the cross product*, the wedge product can be computed without a particular order.

The other property in (**??**) tell us the following. If we have assumed that the three vectors $\mathbf{a}, \mathbf{b}, \mathbf{c}$ span the 3D space (i.e. they do not lay on a plane, than a vector $\mathbf{d}$ cannot be external to them, i.e. its external part is zero.

Bivectors

Let us introduce the bivector $\hat{\mathbf{B}} = \mathbf{b} \wedge \mathbf{c}$. It is apparent that we have that *the external product between a vector and a bivector is commutative*:

$$\mathbf{a} \wedge \hat{\mathbf{B}} = \hat{\mathbf{B}} \wedge \mathbf{a} \,. \tag{33}$$

Once from the matrix d in (**??**) the scalar part is removed, the remaining part corresponds to $\mathbf{w} = \mathbf{a} \cdot (\mathbf{b} \wedge \mathbf{c})$. If we express the components relative to this matrix we have:

$$\begin{aligned} w_x &= -a_3\, b_1\, c_3 - a_2\, b_1\, c_2 + (a_3\, b_3 + a_2\, b_2)\, c_1 \\ w_y &= -a_3\, b_2\, c_3 + (a_3\, b_3 + a_1\, b_1)\, c_2 - a_1\, b_2\, c_1 \\ w_z &= (a_2\, b_2 + a_1\, b_1)\, c_3 - a_2\, b_3\, c_2 - a_1\, b_3\, c_1 \end{aligned} \tag{34}$$

it is noted that they are the same coefficients, apart for a sign, that we would have obtained by performing the vector product $\mathbf{v} = \mathbf{a} \times \mathbf{b} \times \mathbf{c}$.

In fact, in the last part of the code, we perform the triple vector product obtaining:

$$\begin{aligned} v_x &= a_3\,b_1\,c_3 + a_2\,b_1\,c_2 + (-a_3\,b_3 - a_2\,b_2)\,c_1 \\ v_y &= a_3\,b_2\,c_3 + (-a_3\,b_3 - a_1\,b_1)\,c_2 + a_1\,b_2\,c_1 \\ v_z &= (-a_2\,b_2 - a_1\,b_1)\,c_3 + a_2\,b_3\,c_2 + a_1\,b_3\,c_1\,. \end{aligned} \tag{35}$$

By comparing (**??**) with (**??**) it is readily recognized that we have found another important relationship i.e.

$$\mathbf{a}\cdot\mathbf{b}\wedge\mathbf{c}=-\mathbf{a}\times\mathbf{b}\times\mathbf{c} \tag{36}$$

In passing, it is reminded that we have already introduced the relationship $\mathbf{a}\times(\mathbf{b}\times\mathbf{c})=(\mathbf{a}\cdot\mathbf{c})\mathbf{b}-(\mathbf{a}\cdot\mathbf{b})\mathbf{c}$.

We can also compute $\mathbf{a}\cdot\mathbf{b}\wedge\mathbf{c}$ as

$$\mathbf{a}\cdot\mathbf{b}\wedge\mathbf{c}=-\mathbf{a}\times\mathbf{b}\times\mathbf{c}=(\mathbf{a}\cdot\mathbf{b})\mathbf{c}-(\mathbf{a}\cdot\mathbf{c})\mathbf{b} \tag{37}$$

This computation is also performed in the code, where the dot products are obtained as e.g. $\mathbf{a}\cdot\mathbf{b}=\left(\tilde{a}\tilde{b}+\tilde{b}\tilde{a}\right)/2$ and then proceeding with standard matrix multiplication.

So far we have considered the external product of a bivector with a vector (or viceversa) and we have obtained a trivector.

Then we have considered the dot product of a vector with a bivector, obtaining a vector.

It is thus natural to pose the question: what is the result of a product of the type $\mathbf{b} \wedge \mathbf{c} \cdot \mathbf{a}$?

In other words, the dot product of a *bivector* with a vector is commutative or anticommutative? In order to answer to this question we can consider the following code.

`wxMaxima/Pauli_bca.wxm`

which is listed in the following.

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$
load(vect);
/* code name: Pauli_bca.wxm */

declare([a1,a2,a3],scalar)$
declare([b1,b2,b3],scalar)$
declare([c1,c2,c3],scalar)$

print("-----------------------------------------------------------")$
print("Pauli Matrices Definition")$
print("-----------------------------------------------------------")$

%sigma[0] : matrix([1,0],[0,1])$
%sigma[1] : matrix([0,1],[1,0])$
%sigma[2] : matrix([0,-%i],[%i,0])$
%sigma[3] : matrix([1,0],[0,-1])$

/*
print("%sigma[0] =",%sigma[0])$
print("%sigma[1] =",%sigma[1])$
print("%sigma[2] =",%sigma[2])$
print("%sigma[3] =",%sigma[3])$
*/

a : a1 . %sigma[1] + a2 . %sigma[2] + a3 . %sigma[3]$
print("Express vector ",'a," as a Pauli matrix ",a)$
b : b1 . %sigma[1] + b2 . %sigma[2] + b3 . %sigma[3]$
print("Express vector ",'b," as a Pauli matrix ",b)$
c : c1 . %sigma[1] + c2 . %sigma[2] + c3 . %sigma[3]$
print("Express vector ",'c," as a Pauli matrix ",c)$

print("-----------------------------------------------------------")$
print("geometric (Clifford) product bc (between two vectors)    ")$
print("-----------------------------------------------------------")$

print("Find their product")$
bc : ratsimp(b . c);

print("The  scalar part (inner product) is the trace/2")$
sbc : ratsimp((bc[1,1]+bc[2,2])/2);

print("The external product ",'b ,"^", 'c)$
bwc : ratsimp(bc - sbc . %sigma[0]);
print("one recognizes that the above is %i b x c")$
print("-----------------------------------------------------------")$
print(" The following identity holds b ^ c = %i b x c  ")$
print("-----------------------------------------------------------")$
print("Now perform  (b ^ c) a")$
print("-----------------------------------------------------------")$
abwc :  bwc . a ;

print("The  scalar part (inner product) is the trace/2")$
sbc :  factor(ratsimp((abwc[1,1]+abwc[2,2])/2));
print("Note that the above is a trivector (a pseudoscalar)")$
print(" It corresponds to the volume element  b ^ c ^ a")$

print("If we consider the matrix ")$
abc : matrix([a1,a2,a3],[b1,b2,b3],[c1,c2,c3])$
print('abc," = ",abc)$
print(" its determinat is ")$
dabc : determinant(abc);
print("and the determinant multiplied by i is")$
print("equal to  the scalar part, their difference being")$
ratsimp(%i * dabc - sbc);

print("-----------------------------------------------------------")$
print("Therefore we have  b ^ c ^ a = i det(abc)")$
print("-----------------------------------------------------------")$

print("The remaining part corresponds to  (b ^ c) dot a")$
awbwc : ratsimp(abwc - sbc . %sigma[0]);

print("i.e.  a vector with components")$
wx : ratsimp((awbwc[1,1]+awbwc[2,2])/2)$
wx : ratsimp((awbwc[2,1]+awbwc[1,2])/2)$
wy : ratsimp((awbwc[2,1]-awbwc[1,2])/(2*%i))$
wz : ratsimp((awbwc[1,1]-awbwc[2,2])/2)$

print('wx," = ",wx )$
print('wx," = ",wx )$
print('wy," = ",wy )$
print('wz," = ",wz )$


print("-----------------------------------------------------------")$
print(" (a.b)c - (a.c)b  (identity bac- cab = a x b c )  ")$
print("-----------------------------------------------------------")$

print("trying a different approach")$
print("evaluating dot products e.g. a dot b as (ab + ba)/2")$
adb : ratsimp((a . b + b . a)/2);
adc : ratsimp((a . c + c . a)/2);
w : adb . c - adc . b;

ax : ratsimp((w[1,1]+w[2,2])/2)$
ax : ratsimp((w[2,1]+w[1,2])/2)$
ay : ratsimp((w[2,1]-w[1,2])/(2*%i))$
az : ratsimp((w[1,1]-w[2,2])/2)$

print("i.e. a vector with components")$
print('ax," = ",ax )$
print('ax," = ",ax )$
print('ay," = ",ay )$
print('az," = ",az )$

print("Proof that their sum (e.g. wx + ax) vanishes")$
ratsimp(wx+ax);
ratsimp(wy+ay);
ratsimp(wz+az);


print("-----------------------------------------------------------")$
print("Therefore w = a . (b ^ c) = (a.b)c - (a.c)b = -(b ^ c) . a")$
print("The dot product of a vector with a bivector is anticommutative")$
print("-----------------------------------------------------------")$

/*
print("=========================GIBBS=============================")$
e:[a1, a2, a3];
b:[b1, b2, b3];
c:[c1, c2, c3];

print("A X ( B X C ) = ")$
r1 : express(a ^ express(b ^ c));

print("The sum of these coefficients with")$
print("those previously found gives zero ")$
ratsimp(r1[1] + ax);
ratsimp(r1[2] + ay);
ratsimp(r1[3] + az);

print("-----------------------------------------------------------")$
print("Therefore a . (b ^ c) = (a.b)c - (a.c)b = - a x b x c  ")$
print("-----------------------------------------------------------")$
*/
print("end")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```

the dot product for bivectors is anticommutative

The initial part is identical as before, the difference only coming out when we compute $(\mathbf{b} \wedge \mathbf{c})\mathbf{a}$.

The result is a 2x2 matrix with half of the trace that now corresponds to $\mathbf{b} \wedge \mathbf{c} \wedge \mathbf{a}$.

We thus have $\mathbf{b} \wedge \mathbf{c} \wedge \mathbf{a} = \mathbf{a} \wedge \mathbf{b} \wedge \mathbf{c}$.

By subtracting this part (times σ_0) from the matrix $(\mathbf{b} \wedge \mathbf{c})\mathbf{a}$ we obtain the matrix representing $(\mathbf{b} \wedge \mathbf{c}) \cdot \mathbf{a}$ with the components which are the same as in (**??**) but with a minus sign.

Therefore, *the dot product for bivectors is anticommutative* or

$$\mathbf{a} \cdot \mathbf{b} \wedge \mathbf{c} = -\mathbf{b} \wedge \mathbf{c} \cdot \mathbf{a} \,. \tag{38}$$

Bivectors dot and wedge products

Therefore for a bivector $\hat{\mathbf{B}}$ we can write

$$\mathbf{a} \cdot \hat{\mathbf{B}} = \frac{1}{2}\left(\mathbf{a}\hat{\mathbf{B}} - \hat{\mathbf{B}}\mathbf{a}\right) = -\mathbf{a} \times \mathbf{b} \times \mathbf{c} \tag{39}$$

$$\mathbf{a} \wedge \hat{\mathbf{B}} = \frac{1}{2}\left(\mathbf{a}\hat{\mathbf{B}} + \hat{\mathbf{B}}\mathbf{a}\right) = \mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})\, i \tag{40}$$

The last equality has not been shown so far, but the reader can easily prove it.

the vector $\mathbf{B} = \mathbf{b} \times \mathbf{c}$

By introducing the vector $\mathbf{B} = \mathbf{b} \times \mathbf{c}$ we have

$$\hat{\mathbf{B}} = \mathbf{b} \wedge \mathbf{c} = i\,\mathbf{b} \times \mathbf{c} = i\,\mathbf{B} \tag{41}$$

which, by using (**??**, **??**), allows to write

$$\mathbf{a} \cdot \hat{\mathbf{B}} = -\mathbf{a} \times \mathbf{B} \tag{42}$$

$$\mathbf{a} \wedge \hat{\mathbf{B}} = \mathbf{a} \cdot \mathbf{B}\, i \tag{43}$$

Relating classical vector analysis with the Pauli algebra

Equation (**??**) shows that a bivector $\hat{\mathbf{B}}$ can be obtained from a vector $\mathbf{B}$ simply by multiplication times i (and viceversa).

It is also noted that a scalar t when multiplied by i becomes a trivector (and viceversa).

The relationships (**??**, **??**) can be used for relating classical vector analysis with the Pauli algebra.

Vector analysis with Pauli matrices (summary)

Vector analysis with Pauli matrices

- Three–dimensional vectors can be represented using 2x2 Pauli matrices.

$$\tilde{a} = \begin{pmatrix} a_z & a_x - ia_y \\ a_x + ia_y & -a_z \end{pmatrix} \tag{44}$$

- Multiplication of two matrices corresponding to two vectors give us both the dot product and another new part, the external product. This multiplication corresponds to the geometric or Clifford product.

$$\mathbf{ab} = \mathbf{a} \cdot \mathbf{b} + \mathbf{a} \wedge \mathbf{b} = \tilde{a}\tilde{b}. \tag{45}$$

Vector analysis with Pauli matrices

- dot product between two vectors is commutative:

$$\mathbf{a} \cdot \mathbf{b} = \mathbf{b} \cdot \mathbf{a} = \frac{\tilde{a}b + \tilde{b}a}{2}, \tag{46}$$

- the external product between two vectors is anticommutative

$$\mathbf{a} \wedge \mathbf{b} = -\mathbf{b} \wedge \mathbf{a} = \frac{\tilde{a}b - \tilde{b}a}{2}. \tag{47}$$

- the external product, in 3D, is related to the cross product as:

$$\mathbf{a} \wedge \mathbf{b} = i\,\mathbf{a} \times \mathbf{b}. \tag{48}$$

- the external product between two vectors introduces a new subject: *the bivector*.
- the bivector can be expressed either showing the two vector components or a single vector component but multiplied by i

$$\begin{aligned} \sigma_1\sigma_2 &= i\sigma_3 = -\sigma_2\sigma_1 \\ \sigma_2\sigma_3 &= i\sigma_1 = -\sigma_3\sigma_2 \\ \sigma_3\sigma_1 &= i\sigma_2 = -\sigma_1\sigma_3 \,. \end{aligned} \tag{49}$$

- A bivector $\hat{\mathbf{B}} = \mathbf{b} \wedge \mathbf{c}$ can be multiplied by a vector giving rise to a vector and a *trivector*:

$$\mathbf{a}\,\hat{\mathbf{B}} = \mathbf{a} \cdot \hat{\mathbf{B}} + \mathbf{a} \wedge \hat{\mathbf{B}} \tag{50}$$

- the internal product of a vector with a bivector is given by

$$\mathbf{a} \cdot \hat{\mathbf{B}} = \frac{1}{2}\left(\mathbf{a}\hat{\mathbf{B}} - \hat{\mathbf{B}}\mathbf{a}\right) = -\mathbf{a} \times \mathbf{b} \times \mathbf{c} = -\mathbf{a} \times \mathbf{B} \tag{51}$$

- the external product of a vector and a bivector is

$$\mathbf{a} \wedge \hat{\mathbf{B}} = \frac{1}{2}\left(\mathbf{a}\hat{\mathbf{B}} + \hat{\mathbf{B}}\mathbf{a}\right) = \mathbf{a} \cdot (\mathbf{b} \times \mathbf{c})\, i = \mathbf{a} \cdot \mathbf{B}\, i \tag{52}$$

- the dot product of a vector and a bivector is anticommutative and it is related to the cross product as:

$$\mathbf{a} \cdot \mathbf{b} \wedge \mathbf{c} = -\mathbf{a} \times \mathbf{b} \times \mathbf{c} = (\mathbf{a} \cdot \mathbf{b})\mathbf{c} - (\mathbf{a} \cdot \mathbf{c})\mathbf{b} = -\mathbf{b} \wedge \mathbf{c} \cdot \mathbf{a} \tag{53}$$

Space description

It is noted that elements three-dimensional space are described by eight numbers (i.e. a complex 2x2 matrix). In particular they are:

- one *scalar* (σ_0). Grade 0
- 3 basis *vectors* ($\sigma_1, \sigma_2, \sigma_3$) i.e. three directions. Grade 1
- 3 basis *bivectors* ($\sigma_1\sigma_2, \sigma_1\sigma_3, \sigma_2\sigma_3$). Grade 2
- one *pseudoscalar* ($\sigma_1\,\sigma_2\,\sigma_3$). Grade 3

All these elements are contained in a 2x2 matrix and, similarly to what we do for complex numbers, they can be written together in a *multivector M* as

$$M = a_0 + \mathbf{a} + \hat{\mathbf{B}} + \hat{t} \tag{54}$$

where a_0 is a scalar, $\mathbf{a}$ is a vector, $\hat{\mathbf{B}}$ is a bivector and $\hat{t}$ is a pseudoscalar.

Multivector

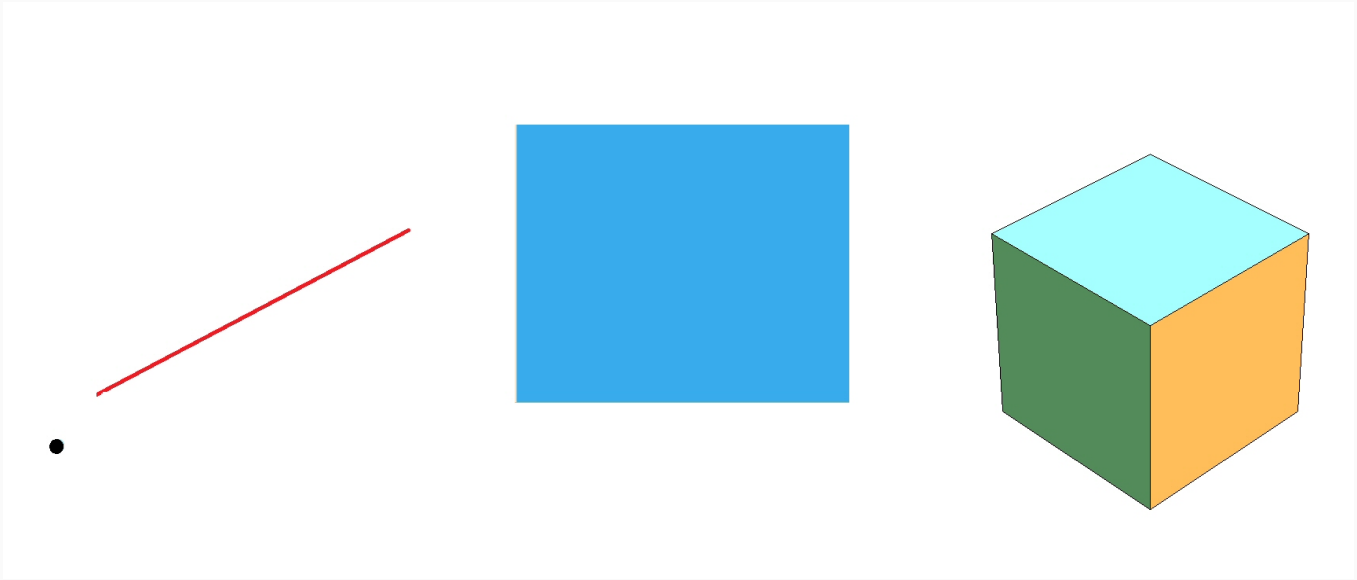

Figure 4: A multivector $M = a_0 + \mathbf{a} + \hat{\mathbf{B}} + \hat{t}$ representing a point, line, area and volume, that can be added, subtracted, multiplied or divided by other multivectors.

Pauli matrix representation of a multivvector

Let us see with more details the Pauli matrix representation of a multivector. The corresponding matrices of the multivector in (**??**) are given next:

$$\begin{aligned}
\tilde{a_0} &= \begin{pmatrix} a_0 & 0 \\ 0 & a_0 \end{pmatrix} \\
\tilde{a} &= \begin{pmatrix} a_3 & a_1 - i\,a_2 \\ i\,a_2 + a_1 & -a_3 \end{pmatrix} \\
\tilde{B} &= \begin{pmatrix} i\,B_3 & B_2 + i\,B_1 \\ i\,B_1 - B_2 & -i\,B_3 \end{pmatrix} \\
\tilde{t} &= \begin{pmatrix} i\,t & 0 \\ 0 & i\,t \end{pmatrix} .
\end{aligned} \tag{55}$$

The matrices in (**??**) can be summed together giving for the multivector M

$$\tilde{M} = \begin{pmatrix} i\,B_3 + i\,t + a_3 + a_0 & B_2 + i\,B_1 - i\,a_2 + a_1 \\ -B_2 + i\,B_1 + i\,a_2 + a_1 & -i\,B_3 + i\,t - a_3 + a_0 \end{pmatrix} . \tag{56}$$

Naturally, for a given Pauli matrix, it is straightforward to retrieve the elements of the different grades.

Let us assume that the matrix in (**??**) is given and we want to retrieve the various elements. It is convenient to extract the real and imaginary part of $\tilde{M}$ as

$$\begin{aligned}
\tilde{M}_r &= \mathrm{Re}\Big\{\tilde{M}\Big\} = \begin{pmatrix} a_3 + a_0 & B_2 + a_1 \\ a_1 - B_2 & a_0 - a_3 \end{pmatrix} \\
\tilde{M}_i &= \mathrm{Im}\Big\{\tilde{M}\Big\} = \begin{pmatrix} B_3 + t & B_1 - a_2 \\ B_1 + a_2 & t - B_3 \end{pmatrix} .
\end{aligned} \tag{57}$$

By inspection, it is seen that we have the following identities:

$$\begin{aligned}
a_0 &= \frac{1}{2}\left(\tilde{M}_{r11} + \tilde{M}_{r22}\right) \\
a_1 &= \frac{1}{2}\left(\tilde{M}_{r12} + \tilde{M}_{r21}\right) \\
a_2 &= \frac{1}{2}\left(\tilde{M}_{i21} - \tilde{M}_{i12}\right) \\
a_3 &= \frac{1}{2}\left(\tilde{M}_{r11} - \tilde{M}_{r22}\right) \\
B_1 &= \frac{1}{2}\left(\tilde{M}_{i21} + \tilde{M}_{i12}\right) \\
B_2 &= \frac{1}{2}\left(\tilde{M}_{r12} - \tilde{M}_{r21}\right) \\
B_3 &= \frac{1}{2}\left(\tilde{M}_{i11} - \tilde{M}_{i22}\right) \\
t &= \frac{1}{2}\left(\tilde{M}_{i11} + \tilde{M}_{i22}\right) .
\end{aligned} \tag{58}$$

The code for converting multivectors into their Pauli matrix equivalent and viceversa is given in the following lines.

```
/* [wxMaxima batch file version 1] [ DO NOT EDIT BY HAND! ]*/
/* [ Created with wxMaxima version 11.08.0 ] */

/* [wxMaxima: input   start ] */
kill(all)$
load(vect);
/* code name: Pauli_multivectors.wxm */

declare([a1,a2,a3],scalar)$
declare([B1,B2,B3],scalar)$
declare([a0,t],scalar)$

%sigma[0] : matrix([1,0],[0,1])$
%sigma[1] : matrix([0,1],[1,0])$
%sigma[2] : matrix([0,-%i],[%i,0])$
%sigma[3] : matrix([1,0],[0,-1])$

A :  a0 . %sigma[0] $
print("Scalar ",'a0," Pauli matrix ",A)$
a : a1 . %sigma[1] + a2 . %sigma[2] + a3 . %sigma[3]$
print("Vector ",'a," Pauli matrix ",a)$
B : %i * (B1 . %sigma[1] + B2 . %sigma[2] + B3 . %sigma[3])$
B : ratsimp(B)$
print("Bivector ",'B," Pauli matrix ",B)$
T : %i * t . %sigma[0] $
print("Pseudoscalar ",'t," Pauli matrix ",T)$

print("Multivector M")$
M : A + a + B + T ;

print("------------------------------------------------------------")$
print("Retrieving the elements of the multivector")$
print("for a given Pauli matrix")$
print("------------------------------------------------------------")$
print("real part")$
Mr : realpart(M);
print("imaginary part")$
Mi : imagpart(M);

a0r : 1/2*(Mr[1,1]+Mr[2,2])$
print('a0r," = ",a0r)$
a1r : 1/2*(Mr[1,2]+Mr[2,1])$
print('a1r," = ",a1r)$
a2r : 1/2*(Mi[2,1]-Mi[1,2])$
print('a2r," = ",a2r)$
a3r : 1/2*(Mr[1,1]-Mr[2,2])$
print('a3r," = ",a3r)$

B1r : 1/2*(Mi[1,2]+Mi[2,1])$
print('B1r," = ",B1r)$
B2r : 1/2*(-Mr[2,1]+Mr[1,2])$
print('B2r," = ",B2r)$
B3r : 1/2*(Mi[1,1]-Mi[2,2])$
print('B3r," = ",B3r)$
t0r : 1/2*(Mi[1,1]+Mi[2,2])$
print('t0r," = ",t0r)$

print("end")$
/* [wxMaxima: input   end   ] */

/* Maxima can't load/batch files which end with a comment! */
"Created with wxMaxima"$
```