A New Number Theory-Algebra Analysis II

Sonaglioni L*
Free Professionist, Italy

Abstract
The basis of this quaternions algebra. The problem of the \( j \cdot k \) product. 3d (and 4d) product and division in algebraic form; also, the algebraic forms of the product and of the division are differentiable. Questions about the possibility of extend this algebra to more dimensions.

Keywords: Quaternions; Operator theory; Algebra; Tensor methods

Three-Dimensions
A recent publication [1] has extended the concepts of the sum and of the product for 3d-4d numbers as new quaternions. The sum and the product, as defined, are commutative.

Paper [1] implicitly gave the definitions of the norm (or modulus) of a 3d number, of the inverse of a 3d number, and of the conjugate.

Figure 1 gives a 3d space representation; the tern \((1, j, k)\) must be considered a tern of orthogonal unit vectors. \(1\) is the real unity and can be omitted in the symbolic calculus; so for the 3d space we can write:

\[
\begin{align*}
\mathbf{s} &= x + j \cdot y + k \cdot z \\
\mathbf{\bar{s}} &= x - j \cdot y - k \cdot z \\
|\mathbf{s}| &= \sqrt{x^2 + y^2 + z^2} \\
\frac{1}{s} &= s \cdot \frac{\mathbf{\bar{s}}}{|s|^2} = \mathbf{\bar{s}} \\
\sigma \cdot s &= \sigma \cdot x + j \cdot \sigma \cdot y + k \cdot \sigma \cdot z \quad \sigma \in \mathbb{R}
\end{align*}
\]

the scalar product and the vector product are also well defined (see code 3d - 2.4g in appendix of paper [2]).

3d scalar product:
\[
s_a \cdot s_b = x_a \cdot x_b + y_a \cdot y_b + z_a \cdot z_b
\]

3d vector product:
\[
s_a \times s_b = \begin{vmatrix} 1 & j & k \\ x_a & y_a & z_a \\ x_b & y_b & z_b \end{vmatrix} [\text{operative formula}]
\]
in algebraic form:
\[
s_a \times s_b = (y_a \cdot z_b - z_a \cdot y_b) + j \cdot (z_a \cdot x_b - x_a \cdot z_b) + k \cdot (x_a \cdot y_b - y_a \cdot x_b)
\]
So, we have the same symbolic of the standard 2d complex numbers.

Paper [2] analyzed some aspects of this algebra, we have seen that this algebra is not distributive, and that this produces some limitations in derivatives and integrals, also we have seen the extended definitions of functions such as \(\sin(s)\) and \(\cos(s)\) may be meaningless.

The problem is because this 3d space is a curved space, the transformations that permit to define the product as a commutative product, are not linear.

Someone could object that the algebraic definition of the \( j \cdot k \) product, in paper [1], is undefined (in polar notation is defined and it is differentiable); in 1843 William Rowan Hamilton has defined the \( j \cdot k \) product in an algebraic form but, with that definition, Hamilton created a non-commutative algebra.

I try now to give an answer about the generic algebraic definition of the product between two 3d numbers as defined in paper [1].

Given:
\[
s_a = x_a + j \cdot y_a + k \cdot z_a
\]

and

*Corresponding author: Sonaglioni L, Free Professionist, Italy, Tel: 388-0579470; E-mail: luca.sonaglioni@hotmail.com

Received February 18, 2016; Accepted February 27, 2016; Published March 03, 2016

Citation: Sonaglioni L (2016) A New Number Theory-Algebra Analysis II. J Appl Comput Math 5: 289. doi:10.4172/2168-9679.1000289

Copyright: © 2016 Sonaglioni L. This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.
\[ s_i = x_i + j \cdot y_i + \tilde{k} \cdot z_i \]

let us develop the algebraic form of the product \( s' = s_a \cdot s_b \)

For a generic \( s \) (3d) number we can write:

\[
\begin{align*}
  r &= \sqrt{x^2 + y^2 + z^2} \\
  c &= x^2 + y^2
\end{align*}
\]

if \( r \neq 0 \) then

\[
\sin(\beta) = \frac{z}{r} ; \quad \cos(\beta) = \frac{1 - \frac{z^2}{c}}{c}
\]

if \( c \neq 0 \) then

\[
\sin(\alpha) = \frac{y}{c} ; \quad \cos(\alpha) = \frac{x}{c}
\]

if \( c=0 \) then \( \alpha = 0 \) if \( x=0 \), \( \beta = \frac{\pi}{2} \cdot \text{sign}(z) \) \( x=0 \), \( y=0 \)

so in this case:

\[
\begin{align*}
  x' &= r_a \cdot r_b \cdot \cos(\beta_a + \beta_b) \cdot \cos(\alpha_a + \alpha_b) \\
  y' &= r_a \cdot r_b \cdot \cos(\beta_a + \beta_b) \cdot \sin(\alpha_a + \alpha_b) \\
  z' &= r_a \cdot r_b \cdot \sin(\beta_a + \beta_b)
\end{align*}
\]

Definitions:

Let us develop the algebraic form of the product \( s' = s_a \cdot s_b \)

so in this case:

\[
\begin{align*}
  \sin(\alpha_a + \alpha_b) &= \frac{y_a}{c_a} \\
  \sin(\beta_a + \beta_b) &= \frac{z_a}{r_a} \\
  \cos(\beta_a + \beta_b) &= \text{sign}(z_a) \cdot \cos(\beta_a) = \text{sign}(z_a) \cdot \frac{c_a}{r_a}
\end{align*}
\]

because in this case

\[
r'_a = \sqrt{x'_a^2 + y'_a^2 + z'_a^2}
\]

it can be observed that

\[
x'_a = -r'_a \cdot \text{sign}(z_a) \cdot \frac{x_a}{c_a} \\
y'_a = -r'_a \cdot \text{sign}(z_a) \cdot \frac{y_a}{c_a} \\
z'_a = r'_a \cdot \text{sign}(z_a) \cdot \frac{z_a}{c_a}
\]

so:

\[
\begin{align*}
  x' &= -r_a \cdot r_b \cdot \text{sign}(z_a) \cdot \frac{x_a}{c_a} \\
  y' &= -r_a \cdot r_b \cdot \text{sign}(z_a) \cdot \frac{y_a}{c_a} \\
  z' &= r_a \cdot r_b \cdot \text{sign}(z_a) \cdot \frac{z_a}{c_a}
\end{align*}
\]

For a generic case:

\[
\beta_a = \frac{\pi}{2} \cdot \text{sign}(z_a)
\]

so:

\[
\begin{align*}
  x' &= -z_a \cdot \frac{x_a}{c_a} \\
  y' &= 0 \\
  z' &= z_a \cdot \frac{c_a}{r_a}
\end{align*}
\]

(4) If \( c = 0 \) and \( c = 0 \) then \( \alpha = 0 \), \( \beta = \frac{\pi}{2} \cdot \text{sign}(z_a) \)

in this case

\[
\begin{align*}
  x' &= -z_a \cdot \frac{x_a}{c_a} \\
  y' &= 0 \\
  z' &= 0
\end{align*}
\]

The generic case (1) of the algebraic product of \( s \cdot s_a \) is differentiable. The other cases are limit case and are differentiable too.

The algebraic form of \( \frac{1}{s} \) is quite simple:

given:

\[
\begin{align*}
  s &= x + j \cdot y + \tilde{k} \cdot z \\
  r &= \sqrt{x^2 + y^2 + z^2} \\
  s' &= \frac{1}{r} \left( x - j \cdot y - \tilde{k} \cdot z \right)
\end{align*}
\]

and it is differentiable.

Now we can define the algebraic form of the division:

\[
\frac{s}{s_b} = \frac{\sqrt{x^2 + y^2 + z^2}}{\sqrt{x_b^2 + y_b^2 + z_b^2}} \\
\]

it can be observed that

\[
\frac{1}{r'_b} = \frac{1}{\left( r_b - j \cdot y_b - \tilde{k} \cdot z_b \right)}
\]
so
\[
\begin{align*}
s' &= \frac{g}{s_b} = \left( x_b + j \cdot y_b + k \cdot z_b \right) \cdot \left( 1 \cdot \left( x_a + j \cdot y_a - k \cdot z_a \right) \right)
\end{align*}
\]
again we have to analyze 4 cases:

(1a) Generic case: \( c \neq 0, c' \neq 0 \)
\[
\begin{align*}
x' &= \frac{1}{r_b} \left( c_a \cdot c' + z_a \cdot z_b \right) \cdot \left( x_a \cdot x_b + y_a \cdot y_b \right) \cdot c_b \cdot c_b
\end{align*}
\]
\[
\begin{align*}
y' &= \frac{1}{r_b} \left( c_a \cdot c' + z_a \cdot z_b \right) \cdot \left( -x_a \cdot y_b + y_a \cdot x_b \right) \cdot c_b \cdot c_b
\end{align*}
\]
\[
\begin{align*}
z' &= \frac{1}{r_b} \left( -c_a \cdot z_b + z_a \cdot c_b \right)
\end{align*}
\]

(2a) If \( c = 0, c' \neq 0 \)
\[
\begin{align*}
x' &= \frac{1}{r_b} \cdot z_b \cdot z_b \cdot x_b \cdot c_b
\end{align*}
\]
\[
\begin{align*}
y' &= \frac{1}{r_b} \cdot z_b \cdot z_b \cdot y_b \cdot c_b
\end{align*}
\]
\[
\begin{align*}
z' &= \frac{1}{r_b} \cdot z_b \cdot c_b
\end{align*}
\]

(3a) If \( c \neq 0, c' = 0 \)
in this case it can be observed that \( \frac{1}{r_b} \cdot z_b = \text{sign}(z_b) \)
\[
\begin{align*}
x' &= \frac{1}{r_b} \cdot z_b \cdot z_b \cdot x_b \cdot c_b
\end{align*}
\]
\[
\begin{align*}
y' &= \frac{1}{r_b} \cdot z_b \cdot z_b \cdot y_b \cdot c_b
\end{align*}
\]
\[
\begin{align*}
z' &= \frac{1}{r_b} \cdot z_b \cdot c_b \cdot -\text{sign}(z_b) \cdot c_b
\end{align*}
\]

(4a) If \( c = 0, c' = 0 \)
\[
\begin{align*}
x' &= \frac{z_b}{r_b}
\end{align*}
\]
\[
\begin{align*}
y' &= 0
\end{align*}
\]
\[
\begin{align*}
z' &= 0
\end{align*}
\]

The generic case (1a) of the algebraic form of the division \( \frac{s_a}{s_b} \) is differentiable. The other cases are limit cases, and are differentiable too, in fact \( \frac{s_a}{s_b} \) can be seen as the product of \( s_a \cdot \frac{1}{s_b} \), where \( |s_b| \neq 0 \)

Anyway, the objectionable case (3a) is:
\[
\begin{align*}
x' &= \text{sign}(z_b) \cdot z_b \cdot x_b \cdot c_b
\end{align*}
\]
\[
\begin{align*}
y' &= \text{sign}(z_b) \cdot z_b \cdot y_b \cdot c_b
\end{align*}
\]
\[
\begin{align*}
z' &= -\text{sign}(z_b) \cdot c_b
\end{align*}
\]

The differential depends on \( x, y, z, z' \) and on the sign of \( z_b \); this because we are doing a differential around a fixed \( z_b \) point (North Pole or South Pole of the sphere), where, what matters of \( z_b \), is just the sign of \( z_b \).

The algebraic definition of the \( j \cdot k \) product can be seen as defined by the algebraic analysis of above and, in particular, it is the limit case (3) (see appendix B for the solved code of the algebraic definition of the 3d product and division).

Another consequence of this analysis is that, now, it is possible to try to analyze a generic 2° order (3d) equation:
\[
\begin{align*}
a \cdot x^2 + b \cdot y^2 + c = 0 \iff a \cdot x^2 + b \cdot y^2 + c \cdot \overline{\mathbf{z}} = 0 \quad | \mathbf{z} | \neq 0
\end{align*}
\]
where, in general, \( a, b, \) and \( c \) can be real numbers or 3d numbers.

Because now we have an algebraic differentiable definition of the product and of the division, it is clear that if we have two 3d functions (see paper [2]) such as:
\[
\begin{align*}
f(x) = x_1(z) + j \cdot y_1(z) + k \cdot z_1(z)
\end{align*}
\]
\[
\begin{align*}
f_2(x) = x_2(z) + j \cdot y_2(z) + k \cdot z_2(z)
\end{align*}
\]
Where \( x_1(z), y_1(z), z_1(z) \) and \( x_2(z), y_2(z), z_2(z) \) are all differentiable functions and they give real results, the product:
\[
\begin{align*}
f(z) &= f_1(z) \cdot f_2(z)
\end{align*}
\]
and the division:
\[
\begin{align*}
f(z) &= f_1(z) / f_2(z)
\end{align*}
\]
are differentiable. This was another open argument of paper [2].

Conclusion

The above analysis has shown it is possible to give an algebraic definition of the product and of the division for 3d numbers as defined in paper [1] and that these algebraic definitions are differentiable.

Same algebraic analysis can also be done for 4d product and division, even that it is a bit more complex (see Figure 2 and appendix A).

For a 4d number we can write:
\[
\begin{align*}
s = x + j \cdot y + k \cdot z + h \cdot t
\end{align*}
\]
the conjugate:
\[
\begin{align*}
\overline{s} = x - j \cdot y - k \cdot z - h \cdot t
\end{align*}
\]
and so on for the inverse property, the norm (or modulus) etc.

In paper [2] I gave a proposal generic sum definition.

Figure 2: 4d space representation.
The objectionable point was to assign by default the v3space’ sign set to 1 in the case that \( r'_{a} = r'_{b} \) \( (r'_{a} = \sqrt{x'^{2} + y'^{2} + z'^{2}}) \); also, it could be questionable the generic 4d scalar product definition. This was a mistake.

The solution is much simpler; the \( \gamma \) angle must be treated in the same way of the \( \beta \) angle; \( \beta \) rotates, but in fact, at the end of calculations \( \beta = \frac{\pi}{2} \) (see Figure 2). The same must be done for \( \gamma \), at the end of calculations \( \gamma \) must be reduced to \( \frac{\pi}{2} \).

So the sum in 4d space is the same of the sum in 3d space (see appendix C) and, because the schema for 4d is the same for 3d, it is obvious that this idea can be extended to more dimensions.

There are no problems to extend the scalar product formula to more dimensions; a last consideration can be done for the extended definition of the 4d vector product.

A 3d number (or a 4d number) can be seen as a vector \( (x,y,z) \). Let us consider C as the 3d resulting vector product between two 3d vectors A and B; C can be seen as an orthogonal vector whose length is the area of the parallelogram identified by the two non-parallel vectors A and B, so the D 4d resulting vector product between tree 4d vectors A, B and C can be defined as an orthogonal vector to A, B and C whose length is the volume of the solid identified by the three non-coplanar 4d vectors A, B and C (for simplicity, you can think that A, B and C are 4d numbers whose \( h \) component value is 0).

The D vector as result of 4d vector product of A, B, and C, is given by the following operative formula:

\[
D = \det \begin{bmatrix}
1 & j & k & h \\
x_{a} & y_{a} & z_{a} & t_{a} \\
x_{b} & y_{b} & z_{b} & t_{b} \\
x_{c} & y_{c} & z_{c} & t_{c}
\end{bmatrix}
\]

The formula can be extended to more dimensions. Versus (sign) of D depends on the tern A, B and C, but these are all well-known questions.

**Appendix A: 4d numbers analysis**

Consider a 4d number:

\[
s = x + j \cdot y + k \cdot z + h \cdot t
\]

given:

\[
r = \sqrt{x^{2} + y^{2} + z^{2} + t^{2}}
\]

\[
\sin(\gamma) = \frac{t}{r}
\]

\[
\cos(\gamma) = \sqrt{1 - \frac{t^{2}}{r^{2}}}
\]

\[
r' = \sqrt{x'^{2} + y'^{2} + z'^{2}} \neq 0
\]

\[
\sin(\beta) = \frac{z}{r'}
\]

\[
\cos(\beta) = \sqrt{1 - \frac{z^{2}}{r'^{2}}}
\]

given:

\[
c = \sqrt{x'^{2} + y'^{2}} \neq 0
\]

\[
\sin(\alpha) = \frac{y}{c}
\]

\[
\cos(\alpha) = \frac{x}{c}
\]

\[
\sin(\gamma) = \frac{t}{r}
\]

If \( c = 0 \) then \( \alpha = 0, \beta = \frac{\pi}{2} \cdot \text{sign}(z) \); \( x = y = 0 \)

If \( r' = 0 \), then \( \beta = 0, \alpha = 0, (x=y=z=0) \)

The 4d product between two 4d numbers \( s' = s_{a} \cdot s_{b} \) is:

\[
s' = (x_{a} + j \cdot y_{a} + k \cdot z_{a} + h \cdot t_{a}) \cdot (x_{b} + j \cdot y_{b} + k \cdot z_{b} + h \cdot t_{b})
\]

\[
s' = s_{a} \cdot s_{b} = x' + j \cdot y' + k \cdot z' + h \cdot t'
\]

The result in polar notation is:

\[
R' = r'_{a} \cdot r'_{b} \cdot \cos(\gamma_{a} + \gamma_{b}) - r'_{a} \cdot r'_{b} \cdot t_{a} \cdot t_{b}
\]

\[
t' = r'_{a} \cdot t_{b} \cdot \sin(\gamma_{a} + \gamma_{b})
\]

\[
z' = R' \sin(\beta_{a} + \beta_{b})
\]

\[
x' = R' \cos(\beta_{a} + \beta_{b}) \cdot \cos(\alpha_{a} + \alpha_{b})
\]

Definitions:

\[
r_{a} = \sqrt{x_{a}^{2} + y_{a}^{2} + z_{a}^{2} + t_{a}^{2}}
\]

\[
r'_{a} = \sqrt{x_{a}'^{2} + y_{a}'^{2} + z_{a}'^{2} + t_{a}'^{2}}
\]

\[
r_{b} = \sqrt{x_{b}^{2} + y_{b}^{2} + z_{b}^{2} + t_{b}^{2}}
\]

\[
r'_{b} = \sqrt{x_{b}'^{2} + y_{b}'^{2} + z_{b}'^{2} + t_{b}'^{2}}
\]

\[
\cos(\gamma_{a} + \gamma_{b}) = \frac{r'_{a} \cdot r'_{b} - t_{a} \cdot t_{b}}{r_{a} \cdot r_{b}}
\]

\[
\sin(\gamma_{a} + \gamma_{b}) = \frac{r'_{a} \cdot t_{b} + r'_{b} \cdot t_{a}}{r_{a} \cdot r_{b}}
\]

now we have to analyze 7 cases:

(1) **Generic case:** \( c_{a} \neq 0 \), \( c_{b} \neq 0 \), \( r'_{a} \cdot r'_{b} \neq 0 \)

\[
r'_{a} = \sqrt{x_{a}'^{2} + y_{a}'^{2} + z_{a}'^{2}}
\]

\[
r'_{b} = \sqrt{x_{b}'^{2} + y_{b}'^{2} + z_{b}'^{2}}
\]

\[
R' = [r'_{a} \cdot r'_{b} - t_{a} \cdot t_{b}]\]

\[
\cos(\beta_{a} + \beta_{b}) = \frac{1}{r'_{a} \cdot r'_{b}} (c_{a} \cdot c_{b} - z_{a} \cdot z_{b})
\]

\[
\cos(\alpha_{a} + \alpha_{b}) = \frac{(x_{a} \cdot x_{b} - y_{a} \cdot y_{b})}{c_{a} \cdot c_{b}}
\]

\[
\sin(\alpha_{a} + \alpha_{b}) = \frac{(x_{a} \cdot y_{b} + y_{a} \cdot x_{b})}{c_{a} \cdot c_{b}}
\]

\[
\sin(\beta_{a} + \beta_{b}) = \frac{1}{r'_{a} \cdot r'_{b}} (c_{a} \cdot z_{b} + z_{a} \cdot c_{b})
\]

\[
x' = \frac{R'}{r'_{a} \cdot r'_{b}} (c_{a} \cdot c_{b} - z_{a} \cdot z_{b}) \cdot (x_{a} \cdot y_{b} - y_{a} \cdot x_{b})
\]
\[
y' = \frac{R'}{r' a r' b} (c_a - z_a - z_b) \left( x_a \cdot y_a + y_a \cdot x_b \right) c_a \cdot c_b
\]
\[
z' = \frac{R'}{r' a r' b} (c_a - z_a + z_c \cdot c_b)
\]
\[
t' = r' a \cdot r' b \cdot \sin(\gamma_a + \gamma_b) = r' a \cdot t_a + r' b \cdot t_b
\]

(2) If \( c_a = 0, \ c_b \neq 0, \ r' a, r' b \neq 0 \) then \( \alpha_a = 0 \)
\[
\beta_a = \frac{\pi}{2} \cdot \text{sign}(z_a), \quad (x_a = 0, y_a = 0)
\]
\[
x' = -\frac{R'}{r' a r' b} \cdot z_a \cdot z_b \cdot \frac{x_b}{c_b}
\]
\[
y' = -\frac{R'}{r' a r' b} \cdot z_a \cdot z_b \cdot \frac{y_b}{c_b}
\]
\[
z' = \frac{R'}{r' a r' b} \cdot z_a \cdot c_b
\]
\[
t' = r' a \cdot t_b + r' b \cdot t_a
\]

(3) If \( c_a \neq 0, \ c_b = 0, \ r' a, r' b \neq 0 \) then \( \alpha_b = 0 \)
\[
\beta_b = \frac{\pi}{2} \cdot \text{sign}(z_b), \quad (x_b = 0, y_b = 0)
\]
\[
x' = -\frac{R'}{r' a r' b} \cdot z_a \cdot z_b \cdot \frac{x_a}{c_a}
\]
\[
y' = -\frac{R'}{r' a r' b} \cdot z_a \cdot z_b \cdot \frac{y_a}{c_a}
\]
\[
z' = \frac{R'}{r' a r' b} \cdot z_b \cdot c_a
\]
\[
t' = r' a \cdot t_b + r' b \cdot t_a
\]

(4) If \( c_a = 0 \) and \( c_b = 0 \) then \( \alpha_a = 0, \ \alpha_b = 0 \)
\[
\beta_a = \frac{\pi}{2} \cdot \text{sign}(z_a), \quad \beta_b = \frac{\pi}{2} \cdot \text{sign}(z_b)
\]
\[
x_a = 0 \quad y_a = 0
\]
\[
\text{So} \quad \beta_a + \beta_b = 0 \quad \text{or} \quad \pm \pi
\]

In this case note that \( r' a \cdot r' b = z_a \cdot z_b \)
\[
x' = -\frac{R'}{r' a r' b} \cdot z_a \cdot z_b \cdot \cos(\beta_a + \beta_b)
\]
\[
y' = 0
\]
\[
z' = 0
\]
\[
t' = r' a \cdot t_b + r' b \cdot t_a
\]

(5) If \( r' a \cdot r' b = 0 \) then \( \alpha_a = 0, \ \beta_a = 0 \)
\[
x' = R' \cdot \cos(\beta_b) = \frac{R'}{r' b} \cdot x_b
\]
\[
y' = R' \cdot \cos(\beta_b) \cdot \sin(\alpha_a) = \frac{R'}{r' b} \cdot y_b
\]
\[
z' = R' \cdot \sin(\beta_b) = \frac{R'}{r' b} \cdot z_b
\]

\[
r' a = \sqrt{x_a^2 + y_a^2 + z_a^2 + t_a^2}
\]
\[
r' b = \sqrt{x_b^2 + y_b^2 + z_b^2 + t_b^2}
\]

\[
r' a = \sqrt{x_a^2 + y_a^2 + z_a^2 + t_a^2}
\]
\[
r' b = \sqrt{x_b^2 + y_b^2 + z_b^2 + t_b^2}
\]

\[
\frac{1}{s} = \frac{1}{r'^a} \left( (x_b - j \cdot y_b - k \cdot z_b - l \cdot t_b) \right)
\]
\[
\frac{z_a}{s_b} = \left( x_a + j \cdot y_a + k \cdot z_a + l \cdot t_a \right) - \left( x_b - j \cdot y_b - k \cdot z_b - l \cdot t_b \right)
\]

Also here we have to analyze \( 7 \) cases:

(1a) Generic case: \( c_a \neq 0, \ c_b \neq 0; \ r' a \cdot r' b \neq 0 \)
\[
r' a = \sqrt{x_a^2 + y_a^2 + z_a^2}
\]
\[
r' b = \sqrt{x_b^2 + y_b^2 + z_b^2}
\]
\[ R' = \left| r'_a r'_b + t_a \cdot t_b \right| \]
\[ x' = \frac{1}{t_b^2} \cdot \frac{R'}{r'_a r'_b} \cdot \left( c_a \cdot c_b + z_a \cdot z_b \right) \cdot \left( x_a - x_b + y_a \cdot y_b \right) / c_a \cdot c_b \]
\[ y' = \frac{1}{t_b^2} \cdot \frac{R'}{r'_a r'_b} \cdot \left( c_a \cdot c_b + z_a \cdot z_b \right) \cdot \left( -x_a + y_b + x_a \cdot y_b \right) / c_a \cdot c_b \]
\[ z' = \frac{1}{t_b^2} \cdot \frac{R'}{r'_a r'_b} \cdot \left( -c_a \cdot z_b + z_a \cdot c_b \right) \]
\[ t' = \frac{1}{t_b^2} \left( -r'_a t_b + r'_b t_a \right) \]

\[(2a) \text{ If } c_a = 0, c_b \neq 0, r'_a r'_b \neq 0 \text{ then } \alpha_a = 0 \]
\[ \beta_a = \frac{\pi}{2} \cdot \text{sign}(z_a); \left( x_a = 0, y_a = 0 \right) \)

\[(3a) \text{ If } c_a \neq 0, c_b = 0, r'_a r'_b \neq 0 \text{ then } \alpha_b = 0 \]
\[ \beta_b = \frac{\pi}{2} \cdot \text{sign}(z_b); \left( x_b = 0, y_b = 0 \right) \]

\[(4a) \text{ If } c_a = 0 \text{ and } c_b = 0, r'_a r'_b \neq 0 \text{ then } \alpha_a = \alpha_b = 0 \]
\[ \beta_a = \frac{\pi}{2} \cdot \text{sign}(z_a), \beta_b = \frac{\pi}{2} \cdot \text{sign}(z_b) \]
\[ \left( x_a = 0, y_a = 0 \right), \left( x_b = 0, y_b = 0 \right) \]

So \( \beta_a - \beta_b = 0 \) or \( \pm \pi \)
\[ x' = \frac{1}{t_b^2} \cdot R' \cdot \text{sign}(z_a) \cdot z_h \]
\[ y' = 0 \]
\[ z' = 0 \]
\[ t' = \frac{1}{t_b^2} \left( -r'_a t_b + r'_b t_a \right) \]

\[(5a) \text{ If } r'_a r'_b = 0; \ r'_a \neq 0; \ c_b = 0; \ \alpha_a = 0; \ \beta_a = 0 \]
\[ x' = \frac{1}{t_b^2} \cdot \frac{R'}{r'_b} \cdot x_b \]
\[ y' = \frac{1}{t_b^2} \cdot \frac{R'}{r'_b} \cdot y_b \]
\[ z' = \frac{1}{t_b^2} \cdot \frac{R'}{r'_b} \cdot z_b \]
\[ t' = \frac{1}{t_b^2} \left( -r'_a t_b + r'_b t_a \right) \]

\[(6a) \text{ If } r'_a r'_b = 0; \ r'_a \neq 0; \ c_a = 0; \ \alpha_b = 0; \ \beta_b = 0 \]
\[ x' = \frac{1}{t_b^2} \cdot \frac{R'}{r'_a} \cdot x_a \]
\[ y' = \frac{1}{t_b^2} \cdot \frac{R'}{r'_a} \cdot y_a \]
\[ z' = \frac{1}{t_b^2} \cdot \frac{R'}{r'_a} \cdot z_a \]
\[ t' = \frac{1}{t_b^2} \left( -r'_a t_b + r'_b t_a \right) \]

\[(7a) \text{ If } r'_a r'_b = 0; \ \alpha_a = \alpha_b = 0 \text{ and } \beta_a = \beta_b = 0 \text{ in this case} \]
\[ \gamma_a - \gamma_b = 0 \text{ or } \pm \pi \]
\[ x' = \left| \frac{t_a}{t_b} \right| \]
\[ y' = 0 \]
\[ z' = 0 \]
\[ t' = 0 \]

The generic case \((1a)\) of the 4d algebraic division \(s_3/s_4\) is differentiable. The other cases are limit case and are also differentiable.

Limit case \((4a)\) may be an objectionable limit case, but, again, is the same questionable problem we have seen above for the division in 3d; the differential depends on the \(z_a\) and \(z_b\) sign.

Appendix B: 3d core visual basic source code

"reference to the code 3d-2.4g in appendix of paper [2]

"The algebraic product

Function MulA_3d(a As Complex3d, b As Complex3d) As Complex3d

Dim Ca As Double, Cb As Double, R As Complex3d
If Near0(a.R) = 0 Or Near0(b.R) = 0 Then Go To Set_To_Zero
Ca = Sqr(a.X ^ 2 + a.Y ^ 2)
Cb = Sqr(b.X ^ 2 + b.Y ^ 2)
If Near0(Ca) <> 0 And Near0(Cb) <> 0 Then

'generic case
R.X = (Ca * Pb - a.Z * b.Z) * (a.X * b.Y - a.Y * b.X) / (Ca * Pb)  
R.Y = (Ca * Pb - a.Z * b.Z) * (a.X * b.Y + a.Y * b.X) / (Ca * Pb)  
R.Z = (Ca * Pb + Cb * a.Z)  
GoTo To_End

End If

If Near0(Ca) = 0 And Near0(Cb) <> 0 Then  
R.X = -a.Z * b.Z * b.X / Cb  
R.Y = -a.Z * b.Z * b.Y / Cb  
R.Z = a.Z * Cb  
GoTo To_End
End If

If Near0(Cb) = 0 And Near0(Ca) <> 0 Then  
R.X = -a.Z * b.Z * a.X / Ca  
R.Y = -a.Z * b.Z * a.Y / Ca  
R.Z = b.Z * Ca  
GoTo To_End
End If

R.X = -a.Z * b.Z  
R.Y = 0  
R.Z = 0

To_End:
Calc_Vector_Notation R 'reference to 3d sub code...
MulA_3d = R
Exit Function

Set_To_Zero:
R.X = 0  
R.Y = 0  
R.Z = 0  
R.Alfa = 0  
R.Beta = 0  
MulA_3d = R
End Function

'The algebraic division
Function DivA_3d(a As Complex3d, b As Complex3d) As Complex3d
Dim Ca As Double, Cb As Double, R As Complex3d, Rb As Double
If Near0(a.R) = 0 Or Near0(b.R) = 0 Then GoTo Set_To_Zero
Rb = 1 / Sqr(b.X ^ 2 + b.Y ^ 2 + b.Z ^ 2)
Ca = Sqr(a.X ^ 2 + a.Y ^ 2)
Cb = Sqr(b.X ^ 2 + b.Y ^ 2)

If Near0(Ca) <> 0 And Near0(Cb) <> 0 Then  
'generic case
R.X = Rb ^ 2 * (Ca * Cb + a.Z * b.Z) * (a.X * b.X + a.Y * b.Y) / (Ca * Cb)
R.Y = Rb ^ 2 * (Ca * Cb + a.Z * b.Z) * (-a.X * b.Y + a.Y * b.X) / (Ca * Cb)
R.Z = Rb ^ 2 * (-Ca * b.Z + Cb * a.Z)
GoTo To_End
End If

If Near0(Ca) = 0 And Near0(Cb) <> 0 Then  
R.X = Rb ^ 2 * a.Z * b.Z * b.X / Cb  
R.Y = -Rb ^ 2 * a.Z * b.Z * b.Y / Cb  
R.Z = Rb ^ 2 * (-a.Z * b.Z + Cb * a.Z)
GoTo To_End
End If

If Near0(Cb) = 0 And Near0(Ca) <> 0 Then
R.X = Sgn(b.Z) * a.Z * a.X / Ca  
R.Y = Sgn(b.Z) * a.Z * a.Y / Ca  
R.Z = -Sgn(b.Z) * Ca
GoTo To_End
End If

If Near0(Cb) = 0 And Near0(Ca) <> 0 Then
R.X = Sgn(b.Z) * a.Z * a.X / Ca  
R.Y = Sgn(b.Z) * a.Z * a.Y / Ca  
R.Z = -Sgn(b.Z) * Ca
GoTo To_End
End If

To_End:
Calc_Vector_Notation R 'reference to 3d sub code...
MulA_3d = R
Exit Function

Set_To_Zero:
R.X = 0
R.Y = 0  
R.Z = 0  
R.R = 0  
R.Alfa = 0  
R.Beta = 0  
DivA_3d = R  
End Function

Appendix C: 4d core visual basic source code.
Option Compare Database  
Option Explicit  
'-----------------------------------  
' CORE 4d ALGEBRA  
' V2.9 OPTIMIZED  
'-----------------------------------  
'Public Const Pi = 3.14159265358979  
'----------------------------------------------  
'AVOID THE USE OF SMALL NUMBER IN SIMULATION (OR  
VERY BIG NUMBERS)  
'THE PRECISION IS LIMITED, THE MANTISSA HAVE 15  
DIGIT  
'Public Const MaxDigit = 12, AsZero = 10 ^ -12  
'We can round the results of calculus or not  
Private Const Round_Results = True  
'----------------------------------------------  
'The definition of the Complex4d type  
Type Complex4d  
X As Double  
Y As Double  
Z As Double  
T As Double  
R As Double  
Alfa As Double  
Beta As Double  
Gamma As Double  
End Type  

'The initialization number in cartesian notation  
Function Init_Algebraic_4d(X As Double, Y As Double, Z As Double, T As Double) As Complex4d  
Dim R As Complex4d  
R.X = X  
R.Y = Y  
R.Z = Z  
R.T = T  
Calc_Vector_Notation R  
Init_Algebraic_4d = R  
End Function  

'The initialization number in vector notation  
Function Init_Vector_4d(R As Double, Alfa As Double, Beta As Double, Gamma As Double) As Complex4d  
Dim S As Complex4d  
S.R = R  
S.Alfa = Alfa  
S.Beta = Beta  
S.Gamma = Gamma  
To_Algebraic_Notation S  
Init_Vector_4d = S  
End Function  

'The sum A+B  
Function Sum_4d(a As Complex4d, b As Complex4d) As Complex4d  
Dim R As Complex4d  
'R = a.X + b.X  
R.X = a.X + b.X  
R.Y = a.Y + b.Y  
R.Z = a.Z + b.Z  
R.T = a.T + b.T  
Calc_Vector_Notation R  
Sum_4d = R  
End Function  

'The difference A-B  
Function Diff_4d(a As Complex4d, b As Complex4d) As Complex4d  
Dim R As Complex4d  
'R = a.X - b.X  
R.X = a.X - b.X  
R.Y = a.Y - b.Y  
R.Z = a.Z - b.Z  
R.T = a.T - b.T  
Calc_Vector_Notation R  
Diff_4d = R  
End Function  

R.T = a.T - b.T
Calc_Vector_Notation R
Diff_4d = R
End Function

'The Product A*B
Function Mul_4d(a As Complex4d, b As Complex4d) As Complex4d
Dim R As Complex4d
R.R = a.R * b.R
R.Alpha = Modulus(a.Alpha + b.Alpha, 2 * Pi)
R.Beta = Modulus(a.Beta + b.Beta, 2 * Pi)
R.Gamma = Modulus(a.Gamma + b.Gamma, 2 * Pi)
To_Algebric_Notation R
Mul_4d = R
End Function

'The algebric product
Function MulA_4d(a As Complex4d, b As Complex4d) As Complex4d
Dim Ca As Double, Cb As Double, R As Complex4d
Dim Ra1 As Double, Rb1 As Double, R1 As Double, Kx As Double
If Near0(a.R) = 0 Or Near0(b.R) = 0 Then GoTo Set_To_Zero
Ra1 = Sqr(a.X ^ 2 + a.Y ^ 2 + a.Z ^ 2)
Rb1 = Sqr(b.X ^ 2 + b.Y ^ 2 + b.Z ^ 2)

If Near0(Ra1) = 0 And Near0(Rb1) = 0 Then 'x=y=z=0
R.X = Abs(b.T * a.T)
R.Y = 0
R.Z = 0
R.T = 0
GoTo To_End
End If
R1 = Abs(Ra1 * Rb1 - a.T * b.T)

Ca = Sqr(a.X ^ 2 + a.Y ^ 2)
Cb = Sqr(b.X ^ 2 + b.Y ^ 2)

If Near0(Ra1 * Rb1) = 0 Then
If Near0(R1) = 0 Then
Kx = R1 / Rb1
R.X = Kx * b.X
R.Y = Kx * b.Y
R.Z = Kx * b.Z
R.T = Ra1 * b.T + Rb1 * a.T
GoTo To_End
Else
Kx = R1 / Ra1
R.X = Kx * a.X
R.Y = Kx * a.Y
R.Z = Kx * a.Z
R.T = Ra1 * b.T + Rb1 * a.T
GoTo To_End
End If
End If

If Near0(Ca) <> 0 And Near0(Cb) <> 0 Then 'generic case
Kx = R1 / (Ra1 * Rb1)
R.X = Kx * (Ca * Cb - a.Z * b.Z) * (a.X * b.X - a.Y * b.Y) / (Ca * Cb)
R.Y = Kx * (Ca * Cb - a.Z * b.Z) * (a.X * b.Y + a.Y * b.X) / (Ca * Cb)
R.Z = Kx * (Ca * b.Z + Cb * a.Z)
R.T = Ra1 * b.T + Rb1 * a.T
GoTo To_End
End If

If Near0(Ca) = 0 And Near0(Cb) <> 0 Then
Kx = R1 / (Ra1 * Rb1)
R.X = -Kx * a.Z * b.Z * b.X / Cb
R.Y = -Kx * a.Z * b.Z * b.Y / Cb
R.Z = Kx * a.Z * Cb
R.T = Ra1 * b.T + Rb1 * a.T
GoTo To_End
End If

If Near0(Cb) = 0 And Near0(Ca) <> 0 Then
Kx = R1 / (Ra1 * Rb1)
R.X = -Kx * a.Z * b.Z * a.X / Ca
R.Y = -Kx \cdot a.Z \cdot b.Z \cdot a.Y / Ca \\
R.Z = Kx \cdot b.Z \cdot Ca \\
R.T = Ra1 \cdot b.T + Rb1 \cdot a.T \\
GoTo To_End \\
End If \\

If Near0(Cb) = 0 And Near0(Ca) = 0 And Near0(Ra1 * Rb1) <> 0 Then \\
R.X = -R1 \cdot Sgn(a.Z \cdot b.Z) \\
R.Y = 0 \\
R.Z = 0 \\
R.T = Ra1 \cdot b.T + Rb1 \cdot a.T \\
GoTo To_End \\
End If \\

To_End: \\
Calc_Vector_Notation R \\
MulA_4d = R \\
Exit Function \\
Set_To_Zero: \\
R.X = 0 \\
R.Y = 0 \\
R.Z = 0 \\
R.T = 0 \\
R.R = 0 \\
R.Alpha = 0 \\
R.Beta = 0 \\
R.Gamma = 0 \\
MulA_4d = R \\
End Function \\

End Function \\

' The algebraic division \\
Function DivA_4d(a As Complex4d, b As Complex4d) As Complex4d \\
Dim Ca As Double, Cb As Double, R As Complex4d \\
Dim Ra1 As Double, Rb1 As Double, R1 As Double, Rb As Double, Kx As Double \\

If Near0(a.R) = 0 Or Near0(b.R) = 0 Then GoTo Set_To_Zero \\
Rb = 1 / b.R \\
Ra1 = Sqr(a.X ^ 2 + a.Y ^ 2 + a.Z ^ 2) \\
Rb1 = Sqr(b.X ^ 2 + b.Y ^ 2 + b.Z ^ 2) \\

If Near0(Ra1) = 0 And Near0(Rb1) = 0 Then 'x=y=z=0 \\
R.X = Abs(a.T / b.T) \\
R.Y = 0 \\
R.Z = 0 \\
R.T = 0 \\
GoTo To_End \\
End If \\

R1 = Abs(Ra1 * Rb1 + a.T * b.T) \\
Ca = Sqr(a.X ^ 2 + a.Y ^ 2) \\
Cb = Sqr(b.X ^ 2 + b.Y ^ 2) \\

If Near0(Ra1 * Rb1) = 0 Then \\
If Near0(Ra1) = 0 Then \\
Kx = Rb ^ 2 * R1 / Rb1 \\
R.X = Kx ^ b.X \\
R.Y = -Kx ^ b.Y \\
R.Z = -Kx ^ b.Z \\
R.T = Rb ^ 2 * (-Ra1 * b.T + Rb1 * a.T) \\
GoTo To_End \\
Else \\
Kx = Rb ^ 2 * R1 / Ra1 \\
R.X = Kx ^ a.X \\
R.Y = Kx ^ a.Y \\
R.Z = Kx ^ a.Z \\

End Function \\

' The Division A/B \\
Function Div_4d(a As Complex4d, b As Complex4d) As Complex4d \\
Dim R As Complex4d \\
Dim RAs Complex4d \\
R.R = a.R / b.R \\
R.Alpha = Modulus(a.Alpha - b.Alpha, 2 * Pi) \\
R.Beta = Modulus(a.Beta - b.Beta, 2 * Pi) \\
R.Gamma = Modulus(a.Gamma - b.Gamma, 2 * Pi) \\
To_Algebric_Notation R \\
Div_4d = R
R.T = Rb^2 * (-Ra1 * b.T + Rb1 * a.T)
GoTo To_End
End If

End If

If Near0(Ca) <> 0 And Near0(Cb) <> 0 Then
  'generic case
  Kx = Rb^2 * R1 / (Ra1 * Rb1)
  R.X = Kx * (Ca * Ch + a.Z * b.Z) * (a.X * b.X + a.Y * b.Y) / (Ca * Cb)
  R.Y = Kx * (Ca * Ch + a.Z * b.Z) * (-a.X * b.Y + a.Y * b.X) / (Ca * Cb)
  R.Z = Kx * (-Ca * b.Z + Cb * a.Z)
  R.T = Rb^2 * (-Ra1 * b.T + Rb1 * a.T)
  GoTo To_End
End If

If Near0(Ca) = 0 And Near0(Cb) <> 0 Then
  Kx = Rb^2 * R1 / Ra1
  R.X = Kx * a.Z * b.Z * b.X / Cb
  R.Y = -Kx * a.Z * b.Z * b.Y / Cb
  R.Z = Kx * a.Z * Cb
  R.T = Rb^2 * (-Ra1 * b.T + Rb1 * a.T)
  GoTo To_End
End If

If Near0(Cb) = 0 And Near0(Ca) <> 0 Then
  Kx = Rb^2 * R1 / Ra1
  R.X = Kx * a.Z * Sgn(b.Z) * a.X / Ca
  R.Y = Kx * a.Z * Sgn(b.Z) * a.Y / Ca
  R.Z = -Kx * Sgn(b.Z) * Ca
  R.T = Rb^2 * (-Ra1 * b.T + Rb1 * a.T)
  GoTo To_End
End If

If Near0(Cb) = 0 And Near0(Ca) = 0 And Near0(Ra1 * Rb1) <> 0 Then
  R.X = 0
  R.Y = 0
  R.Z = 0
  R.T = 0
  R.R = 0
  R.Alfa = 0
  R.Beta = 0
  R.Gamma = 0
  DivA_4d = R
End If

End Function

Function Inverse_4d(S As Complex4d) As Complex4d
  Dim R As Complex4d
  R.R = 1 / S.R
  R.Alfa = Modulus(-S.Alfa, 2 * Pi)
  R.Beta = Modulus(-S.Beta, 2 * Pi)
  R.Gamma = Modulus(-S.Gamma, 2 * Pi)
  To_Algebric_Notation R
  Inverse_4d = R
End Function

Function S_elev_X_4d(S As Complex4d, X As Double) As Complex4d
  Dim R As Complex4d
  R.R = S.R ^ X
  R.Alfa = Modulus(S.Alfa * X, 2 * Pi)
  R.Beta = Modulus(S.Beta * X, 2 * Pi)
  R.Gamma = Modulus(S.Gamma * X, 2 * Pi)
  To_Algebric_Notation R
  S_elev_X_4d = R
End Function
'Square Root of S
Function Sqr_4d(S As Complex4d) As Complex4d
Dim R As Complex4d
R.R = Sqr(S.R)
R.Alfa = Modulus(S.Alfa / 2, 2 * Pi)
R.Beta = Modulus(S.Beta / 2, 2 * Pi)
R.Gamma = Modulus(S.Gamma / 2, 2 * Pi)
To_Algebraic_Notation R
Sqr_4d = R
End Function

'Rotation and Elongation
Function Rotation_4d(S As Complex4d, dAlfa As Double, dBeta As Double, dGamma As Double, Optional dr As Double = 0) As Complex4d
Dim R As Complex4d
R = S
If Near0(R.R) = 0 And Near0(dr) = 0 Then
Rotation_4d = R
Exit Function
End If
R.R = R.R + dr
R.Alfa = Modulus(S.Alfa + dAlfa, 2 * Pi)
R.Beta = Modulus(S.Beta + dBeta, 2 * Pi)
R.Gamma = Modulus(S.Gamma + dGamma, 2 * Pi)
To_Algebraic_Notation R
Rotation_4d = R
End Function

'Creates ds from a vector S and dAlfa,dBeta and dr
Function Differentiate_Vector_4d(S As Complex4d, dAlfa As Double, dBeta As Double, dGamma As Double, Optional dr As Double = 0) As Complex4d
Dim dx As Double, dy As Double, dz As Double, dt As Double, ds As Complex4d
Dim dr1 As Double, R1 As Double
R1 = Sqr(S.X ^ 2 + S.Y ^ 2 + S.Z ^ 2)
dr1 = dr * Cos(S.Gamma) - S.R * Sin(S.Gamma) * dGamma
dt = dr * Sin(S.Gamma) + S.R * Cos(S.Gamma) * dGamma
dz = dr1 * Sin(S.Beta) + S.R * Cos(S.Beta) * dBeta
dy = dr1 * Cos(S.Beta) * Sin(S.Alfa) - S.R * Sin(S.Beta) * Sin(S.Alfa)
dx = dr1 * Cos(S.Beta) * Cos(S.Alfa) * dAlfa
ds = Init_Algebraic_4d(dx, dy, dz, dt)
End Function

'Scalar product
Function A_V_B_4d(a As Complex4d, b As Complex4d) As Double
A_V_B_4d = a.X * b.X + a.Y * b.Y + a.Z * b.Z + a.T * b.T
End Function

'Versor of S
Function Versor_4d(S As Complex4d) As Complex4d
Dim R As Complex4d, R0 As Double
If Near0(S.R) = 0 Then GoTo Set_To_Zero
R = S
R0 = R.R
R.R = 1
R.X = R.X / R0
R.Y = R.Y / R0
R.Z = R.Z / R0
R.T = R.T / R0
Versor_4d = R
Exit Function
Set_To_Zero:
R.X = 0
R.Y = 0
R.Z = 0
R.T = 0
R.R = 0
R.Alfa = 0
R.Beta = 0
R.Gamma = 0
Versor_4d = R
End Function

'Project A along components on B axes; B new real axes
Function Project_A_on_B_4d(a As Complex4d, b As Complex4d) As Complex4d
Dim dx As Double, dy As Double, dz As Double, dt As Double, ds As Complex4d
Dim dr1 As Double, R1 As Double
R1 = Sqr(S.X ^ 2 + S.Y ^ 2 + S.Z ^ 2)
dr1 = dr * Cos(S.Gamma) - S.R * Sin(S.Gamma) * dGamma
dt = dr * Sin(S.Gamma) + S.R * Cos(S.Gamma) * dGamma
dz = dr1 * Sin(S.Beta) + S.R * Cos(S.Beta) * dBeta
dy = dr1 * Cos(S.Beta) * Sin(S.Alfa) - S.R * Sin(S.Beta) * Sin(S.Alfa)
dx = dr1 * Cos(S.Beta) * Cos(S.Alfa) * dAlfa
ds = Init_Algebraic_4d(dx, dy, dz, dt)
End Function
Dim Wx As Complex4d, Wy As Complex4d, Wz As Complex4d, Wt As Complex4d, R As Complex4d, R0 As Double
Dim X As Double, Y As Double, Z As Double, T As Double
Dim BVx As Double, BVy As Double, BVz As Double, BVt As Double

If Near0(b.R) = 0 Then GoTo Set_To_Zero
If Near0(a.R) = 0 Then GoTo Set_To_Zero

' Versors Wx, Wy and Wz the new base
Wx = Versor_4d(b)

' Optimization
Wy = Init_Algebric_4d(-Wy.Y, Wy.X, 0, 0)
Wy.X = -Wx.Y
Wy.Y = Wx.X
Wy.Z = 0
Wy.T = 0

' Wy = Versor_4d(Wy)
R0 = Sqr(Wy.X ^ 2 + Wy.Y ^ 2 + Wy.Z ^ 2 + Wy.T ^ 2)
If Near0(R0) = 0 Then GoTo Set_To_Zero ' New quaternion is undeterminate
Wy.X = Wy.X / R0
Wy.Y = Wy.Y / R0
Wy.Z = Wy.Z / R0
Wy.T = Wy.T / R0

' Consider Wz as
Wz = A_X_B_3d(Wx, Wy) + T = 0
Wz.X = Wx.Y * Wy.Z - Wx.Z * Wy.Y
Wz.Y = Wx.Z * Wy.X - Wx.X * Wy.Z
Wz.Z = Wx.X * Wy.Y - Wx.Y * Wy.X
Wz.T = 0

' Wt: Take Wx and make it orthogonal respect to T
If Near0(Wx.T) = 0 Then
Wt.X = 0
Wt.Y = 0
End If

Wt.Z = 0
Wt.T = 0

Else
Wt.X = Wx.X
Wt.Y = Wx.Y
Wt.Z = Wx.Z
Wt.T = -(Wx.X ^ 2 + Wx.Y ^ 2 + Wx.Z ^ 2) / Wx.T
End If

' The transformation from Cartesian to Vector Notation
Private Sub Calc_Vector_Notation(S As Complex4d)
    Dim SinGamma As Double, CosGamma As Double, R1 As Double
    Dim SinBeta As Double, CosBeta As Double, SinAlfa As Double,
        CosAlfa As Double

    Check_Algebric_Zero_4d S

    'Calc r
    S.R = Sqr(S.X ^ 2 + S.Y ^ 2 + S.Z ^ 2 + S.T ^ 2)
    If Near0(S.R) = 0 Then GoTo Set_To_Zero

    R1 = Sqr(S.X ^ 2 + S.Y ^ 2 + S.Z ^ 2)
    'Solve Gamma....
    SinGamma = S.T / S.R
    CosGamma = R1 / S.R
    If Round(CosGamma, MaxDigit) = 0 Then '->R1=0; considerate T
        If Round(SinGamma, MaxDigit) = 0 Then GoTo Set_To_Zero 'i.e. T=0, and R1=0
        S.Gamma = Pi / 2 * Sgn(S.T)
        End If

    S.Gamma = ArcSin(SinGamma)
    If Near0(R1) = 0 Then 'pure T vector
        S.X = 0
        S.Y = 0
        S.Z = 0
        S.Alfa = 0
        S.Beta = 0
        S.Gamma = Pi / 2 * Sgn(S.T)
        Exit Sub
    End If

    'Solve Gamma....
    SinGamma = S.T / S.R
    'SinGamma can be <=0

    CosGamma = R1 / S.R
    'CosGamma >=0 always
    If Round(CosGamma, MaxDigit) = 0 Then '->R1=0; considerate T
        If Round(SinGamma, MaxDigit) = 0 Then GoTo Set_To_Zero 'i.e. T=0, and R1=0
        S.Gamma = Pi / 2 * Sgn(S.T)
        End If

    S.Gamma = ArcSin(SinGamma)
    If Near0(R1) = 0 Then 'pure T vector
        S.X = 0
        S.Y = 0
        S.Z = 0
        S.Alfa = 0
        S.Beta = 0
        S.Gamma = Pi / 2 * Sgn(S.T)
        Exit Sub
    End If

    'Solve Beta....
    SinBeta = S.Z / R1
    CosBeta = Sqr(S.X ^ 2 + S.Y ^ 2) / R1
    If Round(CosBeta, MaxDigit) = 0 Then
        S.Beta = Pi / 2 * Sgn(S.Z)
        S.Alfa = 0
        Exit Sub
        End If

    S.Beta = ArcSin(SinBeta)

    'Solve Alfa....
    SinAlfa = S.Y / (R1 * CosBeta)
    CosAlfa = S.X / (R1 * CosBeta)
    If Round(CosAlfa, MaxDigit) = 0 Then
        If Round(SinAlfa, MaxDigit) = 0 Then
            S.Alfa = 0
        Else
            S.Alfa = Pi / 2 * Sgn(S.Y)
        End If
    Else
        S.Alfa = Pi / 2 * Sgn(S.Y)
        Exit If
    End If

    Else
        S.Alfa = ArcSin(SinAlfa)
        If CosAlfa < 0 Then
            If CosAlfa < 0 Then
                S.Alfa = ArcSin(SinAlfa)
                If CosAlfa < 0 Then
                    S.Alfa = ArcSin(SinAlfa)
                    If CosAlfa < 0 Then
                        S.Alfa = ArcSin(SinAlfa)
                        If CosAlfa < 0 Then
                            S.Alfa = ArcSin(SinAlfa)
                            If CosAlfa < 0 Then
                                S.Alfa = ArcSin(SinAlfa)
                                If CosAlfa < 0 Then
                                    S.Alfa = ArcSin(SinAlfa)
                                    If CosAlfa < 0 Then
                                        S.Alfa = ArcSin(SinAlfa)
                                        If CosAlfa < 0 Then
                                            S.Alfa = ArcSin(SinAlfa)
                                            If CosAlfa < 0 Then
                                                S.Alfa = ArcSin(SinAlfa)
                                                If CosAlfa < 0 Then
                                                    S.Alfa = ArcSin(SinAlfa)
                                                    If CosAlfa < 0 Then
                                                        S.Alfa = ArcSin(SinAlfa)
                                                        If CosAlfa < 0 Then
                                                            S.Alfa = ArcSin(SinAlfa)
                                                            If CosAlfa < 0 Then
                                                                S.Alfa = ArcSin(SinAlfa)
                                                                If CosAlfa < 0 Then
                                                                    S.Alfa = ArcSin(SinAlfa)
                                                                    If CosAlfa < 0 Then
                                                                        S.Alfa = ArcSin(SinAlfa)
                                                                        If CosAlfa < 0 Then
                                                                            S.Alfa = ArcSin(SinAlfa)
                                                                            If CosAlfa < 0 Then
                                                                                S.Alfa = ArcSin(SinAlfa)
                                                                                If CosAlfa < 0 Then
                                                                                    S.Alfa = ArcSin(SinAlfa)
                                                                                    If CosAlfa < 0 Then
                                                                                        S.Alfa = ArcSin(SinAlfa)
                                                                                        If CosAlfa < 0 Then
                                                                                            S.Alfa = ArcSin(SinAlfa)
                                                                                            If CosAlfa < 0 Then
                                                                                                S.Alfa = ArcSin(SinAlfa)
                                                                                                If CosAlfa < 0 Then
                                                                                                    S.Alfa = ArcSin(SinAlfa)
                                                                                                    If CosAlfa < 0 Then
                                                                                                        S.Alfa = ArcSin(SinAlfa)
                                                                                                        If CosAlfa < 0 Then
                                                                                                            S.Alfa = ArcSin(SinAlfa)
                                                                                                            If CosAlfa < 0 Then
                                                                                                                S.Alfa = ArcSin(SinAlfa)
                                                                                                                If CosAlfa < 0 Then
                                                                                                                    S.Alfa = ArcSin(SinAlfa)
                                                                                                                    If CosAlfa < 0 Then
                                                                                                                        S.Alfa = ArcSin(SinAlfa)
                                                                                                                        If CosAlfa < 0 Then
                                                                                                                            S.Alfa = ArcSin(SinAlfa)
                                                                                                                            If CosAlfa < 0 Then
                                                                                                                                S.Alfa = ArcSin(SinAlfa)
                                                                                                                                If CosAlfa < 0 Then
                                                                                                                                    S.Alfa = ArcSin(SinAlfa)
                                                                                                                                    If CosAlfa < 0 Then
                                                                                                                                        S.Alfa = ArcSin(SinAlfa)
                                                                                                                                        If CosAlfa < 0 Then
                                                                                                                                            S.Alfa = ArcSin(SinAlfa)
                                                                                                                                            If CosAlfa < 0 Then
                                                                                                                                                S.Alfa = ArcSin(SinAlfa)
                                                                                                                                                If CosAlfa < 0 Then
                                                                                                                                                    S.Alfa = ArcSin(SinAlfa)
                                                                                                                                                    If CosAlfa < 0 Then
                                                                                                                                                        S.Alfa = ArcSin(SinAlfa)
                                                                                                                                                        If CosAlfa < 0 Then
                                                                                                                                                            S.Alfa = ArcSin(SinAlfa)
                                                                                                                                                            If CosAlfa < 0 Then
                                                                                                                                                                S.Alfa = ArcSin(SinAlfa)
                                                                                                                                                                If CosAlfa < 0 Then
                                                                                                                                                                    S.Alfa = ArcSin(SinAlfa)
                                                                                                                                                                    If CosAlfa < 0 Then
                                                                                                                                                                        S.Alfa = ArcSin(SinAlfa)
                                                                                                                                                                        If CosAlfa < 0 Then
                                                                                                                                                                            S.Alfa = ArcSin(SinAlfa)
                                                                                                                                                                            If CosAlfa < 0 Then
                                                                                                                                                                                S.Alfa = ArcSin(SinAlfa)
                                                                                                                                                                                If CosAlfa < 0 Then
                                                                                                                                                                                    S.Alfa = ArcSin(SinAlfa)
                                                                                                                                                                                    If CosAlfa < 0 Then
                                                                                                                                                                                        S.Alfa = ArcSin(SinAlfa)
                                                                                                                                                                                        If CosAlfa < 0 Then
                                                                                                                                                                                            S.Alfa = ArcSin(SinAlfa)
                                                                                                                                                                                            If CosAlfa < 0 Then
                                                                                                                                                                                                S.Alfa = ArcSin(SinAlfa)
                                                                                                                                                                                                If CosAlfa < 0 Then
                                                                                                                                                                                                    S.Alfa = ArcSin(SinAlfa)
                                                                                                                                                                                                    If CosAlfa < 0 Then
                                                                                                                                                                                                        S.Alfa = ArcSin(SinAlfa)
                                                                                                                                        Exit Sub
                                                                                                                                        Set_To_Zero:
                                                                                                                                            S.X = 0
                                                                                                                                            S.Y = 0
                                                                                                                                            S.Z = 0
                                                                                                                                            S.R = 0
                                                                                                                                            S.Alfa = 0
                                                                                                                                            S.Beta = 0
                                                                                                                                            S.Gamma = 0
                                                                                                                                            Exit Sub
                                                                                                                                            S.Beta = ArcSin(SinBeta)
                                                                                                                                            'Solve Alfa....
                                                                                                                                            S.Alfa = Pi / 2 * Sgn(S.Z)
                                                                                                                                            S.Alfa = 0
                                                                                                                                            Exit Sub
                                                                                                                                            End If

        End If
    End If

    Exit Sub
    Set_To_Zero:
        S.X = 0
        S.Y = 0
        S.Z = 0
        S.R = 0
        S.Alfa = 0
        S.Beta = 0
        S.Gamma = 0
        Exit Sub
    End Sub
'THE TRANSFORMATION FROM VECTOR TO CARTESIAN NOTATION

Private Sub To_Algebraic_Notation(S As Complex4d)
    Dim R1 As Double, CosBeta As Double, CosGamma As Double
    If Near0(S.R) = 0 Then GoTo Set_To_Zero

    'Solve X,Y,Z, T
    S.T = S.R * Sin(S.Gamma)
    CosGamma = Cos(S.Gamma)
    If Near0(CosGamma) = 0 Then
        'The Vector is a pure T vector, so
        S.Z = 0
        S.Y = 0
        S.X = 0
        'Alfa and Beta irrelevant, set to 0
        S.Beta = 0
        S.Alfa = 0
        S.Gamma = Pi / 2 * Sgn(S.T)
        Exit Sub
    End If
    R1 = S.R * Abs(CosGamma)

    'Solve Alfa, Beta
    S.Z = R1 * Sin(S.Beta)
    CosBeta = Cos(S.Beta)
    If Near0(CosBeta) = 0 Then
        S.Y = 0
        S.X = 0
        S.Alfa = 0
    Else
        S.Y = R1 * CosBeta * Sin(S.Alfa)
        S.X = R1 * CosBeta * Cos(S.Alfa)
    End If

    Calc_Vector_Notation S
    Exit Sub

Set_To_Zero:
    S.X = 0
    S.Y = 0
    S.Z = 0
    S.T = 0
    S.R = 0
    S.Alfa = 0
    S.Beta = 0
    S.Gamma = 0
End Sub

Private Sub Check_Algebraic_Zero_4d(S As Complex4d)
    S.Z = Near0(S.Z)
    S.Y = Near0(S.Y)
    S.X = Near0(S.X)
    S.T = Near0(S.T)
End Sub

References
1. Sonaglioni L (2015) A New Number Theory. J Appl Computat Math 4: 212.
2. Sonaglioni L (2015) A New Number Theory-Algebra Analysis. J Appl Computat Math 4: 267.
3. Walker MJ (1955) Quaternions as 4-Vectors. Am J Phys 24: 515.
4. Stephenson RJ (1966) Development of Vector Analysis from Quaternions. Am J Phys 34: 194.