Journal of
Applied & Computational Mathematics

ISSN: 2168-9679

Sonaglioni, J Appl Computat Math 2015, 4:2
DOI: 10.4172/2168-9679.1000212

Research Article

Open Access

A New Number Theory

Luca Sonaglioni*

Free Professionist, Italy

Abstract

The article proposes a mathematical method that permits to treat numbers with more than 2 dimensions. The sums and the products must be done by two distinct definitions. Despite this little limitation you have the same algebra of the standard complex numbers. The limitations occur only when you have to really compute the sums and the products between numbers, not in the symbolic algebra.

The product and the sum are commutative.

Keywords: Quaternions; Number Theory; Operator theory; Algebra; Tensor methods

Definitions

Let us consider the 3d space that can be represented by the tern $\vec{u},\vec{v},\vec{w}$ as shown in the Figure 1

The point P can be written as: P= (x,y,z) = $\vec{u}\cdot x+\vec{v}\cdot y+\vec{w}\cdot z$

But also, using the polar notation the point P can be written as P= $r\cdot e^{j\cdot\alpha+k\cdot\beta}$

where $r=\sqrt{x^2+y^2+z^2}$

The operator $e^{k\cdot\beta}$ raises the lying vectors in the plane $\langle\vec{u},\vec{v}\rangle$ of β radians along $\vec{w}$;

The operator $e^{j\cdot\alpha}$ rotates the lying vectors in the plane $\langle\vec{u},\vec{v}\rangle$ of α radians.

Definition of the sum:

1) $P_1+P_2=\vec{u}\cdot(x_1+x_2)+\vec{v}\cdot(y_1+y_2)+\vec{w}\cdot(z_1+z_2)$ [Cartesian notation]

Definition of the product:

2) $P_1\cdot P_2=r_1\cdot e^{j\cdot\alpha_1+k\cdot\beta_1}\cdot r_2\cdot e^{j\cdot\alpha_2+k\cdot\beta_2}=r_1\cdot r_2\cdot e^{j\cdot(\alpha_1+\alpha_2)+k\cdot(\beta_1+\beta_2)}$ [Polar notation]

The Number

$$s=\vec{u}\cdot x+\vec{v}\cdot y+\vec{w}\cdot z\equiv re^{j\alpha+k\beta}$$

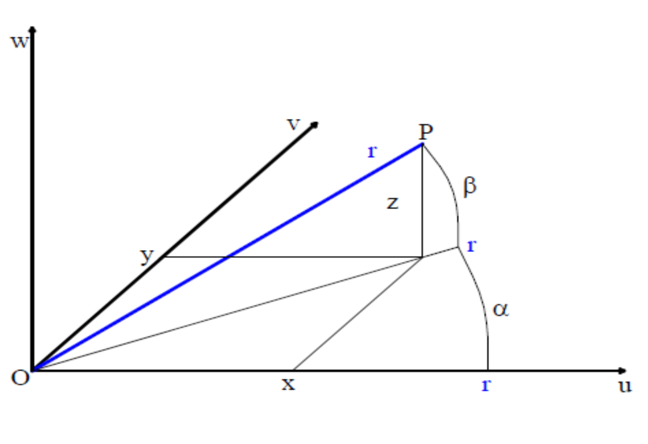

Figure 1: $\vec{u},\vec{v},\vec{w}$ are unit vector in the three-dimensional space.

Behaves as a complex number with three dimensions provided that:

To do the sums it must always be used the definition (1) [Cartesian notation]. To do the products it must always be used the definition (2) [polar notation]. The definition can be extended to 4 dimensions.

The relations between P= (x,y,z) and (r,α,β) are given by the following formulas:

$$3)\quad \begin{aligned} z &= r\cdot\sin(\beta)\\ y &= r\cdot\cos(\beta)\sin(\alpha)\\ x &= r\cdot\cos(\beta)\cos(\alpha)\end{aligned}$$

For $\alpha=0$, the vector is lying in the plane $\langle\vec{u},\vec{w}\rangle$, and the polar notation coincides with a vector in the vertical rotation. The y values for $\alpha=0$ are 0 by default. The transition from one format to another is always possible, because the tern (x,y,z) always and uniquely identifies the tern (r,α,β) through the formulas (3). The methods of symbolic computation are identical to those of the standard complex numbers.

The operations, of calculation, must always take into account the two rules (1) and (2) above for the sums and products. To assess the calculator expressions you can use the Reverse Polish Notation (RPN).

Calculus

The question is: does the calculus work?

Fixing:

$$d\vec{s}=\vec{u}\cdot dx+\vec{v}\cdot dy+\vec{w}\cdot dz$$

The definition above, can't coincides at the infinitesimal to the differential of

$$ds=d\left(r\cdot e^{j\alpha+k\beta}\right)$$

The calculus works using the formulas (3)

$$dz=dr\cdot\sin(\beta)+r\cdot\cos(\beta)\cdot d\beta$$

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

Received February 03, 2015; **Accepted** March 28, 2015; **Published** April 10, 2015

Citation: Sonaglioni L (2015) A New Number Theory. J Appl Computat Math 4: 212. doi:10.4172/2168-9679.1000212

$$dy = dr \cdot \cos(\beta) \cdot \sin(\alpha) - r \cdot \sin(\beta) \cdot \sin(\alpha) \cdot d\beta + r \cdot \cos(\beta) \cdot \cos(\alpha) \cdot d\alpha$$

$$dx = dr \cdot \cos(\beta) \cdot \cos(\alpha) - r \cdot \sin(\beta) \cdot \cos(\alpha) \cdot d\beta - r \cdot \cos(\beta) \cdot \sin(\alpha) \cdot d\alpha$$

Practically the polar notation is useful only to define the concept of the commutative product, necessary for the symbolic operations and for the real calculation of the product between the numbers.

As a last consideration, we can also define $s \cdot s = r^2 \cdot e^{(j\cdot\alpha + k\cdot\beta)\cdot 2}$

and in general: $s^x = r^x \cdot e^{(j\cdot\alpha + k\cdot\beta)\cdot x}, \quad x \in \Re$

4 Dimensions

Let us consider s as a 4 dimension number as defined above:

$$s = \vec{u} \cdot x + \vec{v} \cdot y + \vec{w} \cdot z + \vec{q} \cdot t \equiv r \cdot e^{j\cdot\alpha + k\cdot\beta + h\cdot\lambda}$$

Is it still true the analysis above? The answer is yes, but we have to introduce the operator $e^{\underline{h}\cdot\lambda}$ which raises the vectors in the cube $\langle \vec{u}, \vec{v}, \vec{w} \rangle$ of λ radians along q.

The unity vector $\vec{q}$ is orthogonal to $\vec{u}$, $\vec{v}$ and $\vec{w}$, but it can't be represented graphically; the formulas between the two representations, one Cartesian, the other polar, can be detected from an idea given by the Figure 2.

The formulas:

$$r = \sqrt{x^2 + y^2 + z^2 + t^2} \qquad \alpha \in [0, 2\pi]$$

$$r' = \sqrt{x^2 + y^2 + z^2} = r \cdot |\cos(\gamma)| \qquad \beta \in [0, 2\pi]$$

$$\gamma \in [0, 2\pi]$$

$$t = r \cdot \sin(\gamma)$$

$$z = r' \cdot \sin(\beta)$$

$$y = r' \cdot \cos(\beta)\sin(\alpha)$$

$$x = r' \cdot \cos(\beta)\cos(\alpha)$$

Because r' can't be negative, it is clear that γ must be reduced to $\gamma \in [-\pi/2, \pi/2]$

When $|\gamma| = \pi/2$, z=y=x=0, i.e. a pure $\vec{q}$ vector.

Examples: Calculating the volume of a sphere

We need a little core of the new algebra, see below.

Here a couple of routines (written in Visual Basic) to estimates the volume of a sphere with the algebra above. The more Kloop is high, the more accurate is the estimation.

Sub **SphereA** (R0 as Double, Kloop As Integer)

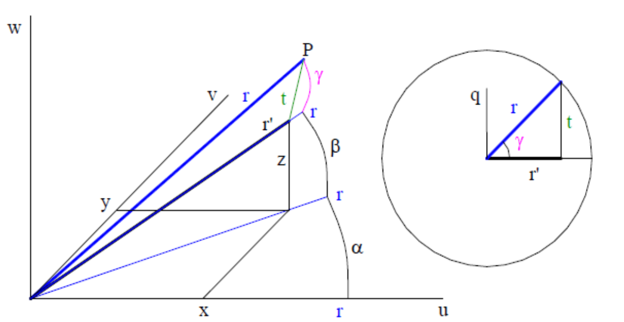

Figure 2: 3D space and original vector.

```
Dim S As Complex3d
Dim dAlfa As Double, dBeta As Double, dr As Double
Dim I As Long, J As Long, K As Long, KLoop As Long
Dim Vol As Double, dv As Double
dAlfa=2 * Pi / KLoop
'Let consider half a sphere
dBeta=Pi / 2 / KLoop
dr=R0 / KLoop
Vol=0
S=Init_Vector(0, 0, 0)
For K=1 To K Loop
For I=1 To K Loop
For J=1 To K Loop
'Vector differential
dv=S.R ^ 2 * dAlfa * Cos(S.Beta) * dBeta * dr
Vol=Vol + dv
S=Rotation_3d(S, dAlfa, 0)
Next J
S=Rotation_3d(S, 0, dBeta)
Next I
S.Beta=0
S=Rotation_3d(S, 0, 0, dr)
Next K
MsgBox "Sphere: estimated Vol: " + Format(2 * Vol) + vbCrLf + "Exact: " + Format(4 / 3 * Pi * R0 ^ 3)
End Sub
Sub SphereB(R0 As Double, Kloop As Integer)
Dim S As Complex3d
Dim dAlfa As Double, dBeta As Double, dr As Double
Dim I As Long, J As Long, K As Long
Dim Vol As Double, dv As Complex3d
dAlfa=2 * Pi / Kloop
dBeta=Pi / 2 / Kloop
dr=R0 / Kloop
Vol=0
S=Init_Vector(0, 0, 0)
For K=1 To Kloop
'Msg "K: " + Format(K) + " di " + Format(Kloop)
For I=1 To Kloop
For J=1 To Kloop
```

```
dv=Differentiate_Vector_3d(S, dAlfa, dBeta, dr)
dv=Project_A_on_B_3d(dv, S)
Vol=Vol + dv.X * dv.Y * dv.Z
S=Rotation_3d(S, dAlfa, 0)
Next J
S=Rotation_3d(S, 0, dBeta)
Next I
S.Beta=0
S=Rotation_3d(S, 0, 0, dr)
DoEvents
Next K
'Msg
MsgBox "Sphere: estimated Vol: " + Format(2 * Vol) + vbCrLf + "Exact: " + Format(4 / 3 * Pi * R0 ^ 3)
End Sub
```

Here the code written in Visual Basic that define the sum, the product's, and s^x…

```
Option Explicit
'-----------------------------------
'  CORE 3d ALGEBRA
'  V2.4d * 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 Complex3d type
Type Complex3d
X As Double
Y As Double
Z As Double
R As Double
Alfa As Double
Beta As Double
End Type
'The initialization number as in an Algebric notation
Function Init_Algebric(X As Double, Y As Double, Z As Double) As Complex3d
Dim R As Complex3d
R.X=X
R.Y=Y
R.Z=Z
Calc_Vector_Notation R
Init_Algebric=R
End Function
'The initialization number as in a Vector notation
Function Init_Vector(R As Double, Alfa As Double, Beta As Double) As Complex3d
Dim S As Complex3d
S.R=R
S.Alfa=Alfa
S.Beta=Beta
To_Algebric_Notation S
Init_Vector=S
End Function
'The Sum A+B
Function Sum_3d (A As Complex3d, B As Complex3d) As Complex3d
Dim R As Complex3d
R.X=A.X + B.X
R.Y=A.Y + B.Y
R.Z=A.Z + B.Z
Calc_Vector_Notation R
Sum_3d=R
End Function
'The Difference A-B
Function Diff_3d (A As Complex3d, B As Complex3d) As Complex3d
Dim R As Complex3d
R.X=A.X - B.X
R.Y=A.Y - B.Y
R.Z=A.Z - B.Z
Calc_Vector_Notation R
Diff_3d=R
End Function
'The Product A*B
Function Mul_3d (A As Complex3d, B As Complex3d) As Complex3d
Dim R As Complex3d, X As Double
R.R=A.R * B.R
```

```
    R.Alfa=Modulus (A.Alfa + B.Alfa, 2 * Pi)
    R.Beta=Modulus (A.Beta + B.Beta, 2 * Pi)
    To_Algebric_Notation R
    Mul_3d=R
    End Function
    'The Division A/B
Function Div_3d (A As Complex3d, B As Complex3d) As Complex3d
    Dim R As Complex3d, X As Double
    'Fai la divisione...
    R.R=A.R / B.R
    R.Alfa=Modulus(A.Alfa - B.Alfa, 2 * Pi)
    R.Beta=Modulus(A.Beta - B.Beta, 2 * Pi)
    To_Algebric_Notation R
    Div_3d=R
    End Function
    'The 1/S
    Function Inverse_3d(S As Complex3d) As Complex3d
    Dim R As Complex3d, X As Double
    R.R=1 / S.R
    R.Alfa=-S.Alfa
    R.Beta=-S.Beta
    To_Algebric_Notation R
    Inverse_3d=R
    End Function
    'S^X; X Real
Function S_elev_X_3d(S As Complex3d, X As Double) As Complex3d
    Dim R As Complex3d
    R.R=S.R ^ X
    If Abs(X) <= 1 Then
    R.Alfa=S.Alfa * X
    R.Beta=S.Beta * X
    Else
    R.Alfa=Modulus(S.Alfa * X, 2 * Pi)
    R.Beta=Modulus(S.Beta * X, 2 * Pi)
    End If
    To_Algebric_Notation R
    S_elev_X_3d=R
    End Function
    'Square Root of S
    Function Sqr_3d(S As Complex3d) As Complex3d
    Dim R As Complex3d
    R.R=Sqr(S.R)
    R.Alfa=S.Alfa / 2
    R.Beta=S.Beta / 2
    To_Algebric_Notation R
    Sqr_3d=R
    End Function
    'Rotation and Elongation
Function Rotation_3d(S As Complex3d, dAlfa As Double, dBeta As Double, Optional dr As Double=0) As Complex3d
    Dim R As Complex3d
    R=S
    If Near0(R.R)=0 And Near0(dr)=0 Then
    Rotation_3d=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)
    To_Algebric_Notation R
    Rotation_3d=R
    End Function
    'Creates ds from a vector S and dAlfa,dBeta and dr
Function Differentiate_Vector_3d(S As Complex3d, dAlfa As Double, dBeta As Double, dr As Double) As Complex3d
    Dim dx As Double, dy As Double, dz As Double, ds As Complex3d
    dz=dr * Sin(S.Beta) + S.R * Cos(S.Beta) * dBeta
    dy=dr * Cos(S.Beta) * Sin(S.Alfa) - S.R * Sin(S.Beta) * Sin(S.Alfa) * dBeta + S.R * Cos(S.Beta) * Cos(S.Alfa) * dAlfa
    dx=dr * Cos(S.Beta) * Cos(S.Alfa) - S.R * Sin(S.Beta) * Cos(S.Alfa) * dBeta - S.R * Cos(S.Beta) * Sin(S.Alfa) * dAlfa
    ds=Init_Algebric(dx, dy, dz)
    Differentiate_Vector_3d=ds
    End Function
    'Internal product
Function A_V_B_3d(A As Complex3d, B As Complex3d) As Double
    A_V_B_3d=A.X * B.X + A.Y * B.Y + A.Z * B.Z
    End Function
    'External product
Function A_X_B_3d(A As Complex3d, B As Complex3d) As Complex3d
    Dim X As Double, Y As Double, Z As Double, R As Complex3d
```

Citation: Sonaglioni L (2015) A New Number Theory. J Appl Computat Math 4: 212. doi:10.4172/2168-9679.1000212


```
X=A.Y * B.Z - A.Z * B.Y
Y=A.Z * B.X - A.X * B.Z
Z=A.X * B.Y - A.Y * B.X
R=Init_Algebric(X, Y, Z)
A_X_B_3d=R
End Function
'Versor of S
Function Versor_3d(S As Complex3d) As Complex3d
Dim R As Complex3d, R0 As Double
If Near0(S.R)=0 Then GoTo Set_To_Zero
R=S
R0=R.R
R.R=1
'---------------------------------
'Optimization
'To_Algebric_Notation R
R.X=R.X / R0
R.Y=R.Y / R0
R.Z=R.Z / R0
Check_Algebric_Zero R
'----------------------------------
Versor_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
   Versor_3d=R
End Function
'Return vector A along components on B axes; B new Real axes
Function Project_A_on_B_3d(A As Complex3d, B As Complex3d) As Complex3d
Dim Wx As Complex3d, Wy As Complex3d, Wz As Complex3d, R As Complex3d, R0 As Double
Dim X As Double, Y As Double, Z As Double
Dim BVx As Double, BVy As Double, BVz 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_3d(B)
'--------------------------------------------------
'      Optimization
'Wy=Init_Algebric_3d(-Wy.Y, Wy.X, 0)
Wy.X=-Wx.Y
Wy.Y=Wx.X
Wy.Z=0
'Wy=Versor_3d(Wy)
R0=Sqr(Wy.X ^ 2 + Wy.Y ^ 2)
If Near0(R0)=0 Then R0=1
Wy.X=Wy.X / R0
Wy.Y=Wy.Y / R0
'--------------------------------------------------
'
'--------------------------------------------------
'consider Wz as
'Wz=A_X_B_3d(Wx, Wy)
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=Versor_3d(Wz)
'--------------------------------------------------
'
'Project A on Wx, Wy, Wz, Wt
BVx=A_V_B_3d(A, Wx)
BVy=A_V_B_3d(A, Wy)
BVz=A_V_B_3d(A, Wz)
R=Init_Algebric(BVx, BVy, BVz)
Project_A_on_B_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
   Project_A_on_B_3d=R
End Function
```

Citation: Sonaglioni L (2015) A New Number Theory. J Appl Computat Math 4: 212. doi:10.4172/2168-9679.1000212

'THE TRASFORMATION FROM ALGEBRIC TO POLAR NOTATION

```
Private Sub Calc_Vector_Notation(S As Complex3d)
Dim SinBeta As Double, CosBeta As Double, SinAlfa As Double, CosAlfa As Double
Check_Algebric_Zero S
'Calc R
S.R=Sqr(S.X ^ 2 + S.Y ^ 2 + S.Z ^ 2)
If Near0(S.R)=0 Then GoTo Set_To_Zero
'Solve Beta....
SinBeta=S.Z / S.R
CosBeta=Sqr(S.X ^ 2 + S.Y ^ 2) / S.R
'CosBeta is always >=0
'SinBeta can be <=0
If Round(CosBeta, MaxDigit)=0 Then
If Round(SinBeta, MaxDigit)=0 Then GoTo Set_To_Zero
S.Beta=Pi / 2 * Sgn(S.Z)
Else
S.Beta=ArcSin(SinBeta)
End If
'Solve Alfa....
If Round(CosBeta, MaxDigit) <> 0 Then
SinAlfa=S.Y / S.R / CosBeta
CosAlfa=S.X / S.R / CosBeta
'CosAlfa can be <=0 ...
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=ArcSin(SinAlfa)
If CosAlfa < 0 Then
'If CosAlfa<0 ... -> Quadrant 2 o quadrant 4
If Near0(S.Alfa) <> 0 Then
S.Alfa=(Pi - Abs(S.Alfa)) * Sgn(S.Y)
Else
S.Alfa=Pi
End If
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
End Sub
```

'THE TRASFORMATION FROM VECTOR TO ALGEBRIC

```
Private Sub To_Algebric_Notation(S As Complex3d)
Dim CosBeta As Double
If Near0(S.R)=0 Then GoTo Set_To_Zero
'Solve X,Y,Z
S.Z=S.R * Sin(S.Beta)
CosBeta=Cos(S.Beta)
If Near0(CosBeta)=0 Then 'a w vector!
S.Y=0
S.X=0
'If CosBeta=0 Alfa is irrelevant
S.Alfa=0
Else
S.Y=S.R * CosBeta * Sin(S.Alfa)
S.X=S.R * CosBeta * Cos(S.Alfa)
End If
Check_Algebric_Zero S
If Near0(S.Alfa)=0 And S.X<0 Then S.Alfa=Pi
If Near0(S.Z)=0 And S.Beta<>0 Then S.Beta=0
Exit Sub
Set_To_Zero:
S.X=0
S.Y=0
S.Z=0
S.R=0
S.Alfa=0
S.Beta=0
End Sub
Private Sub Check_Algebric_Zero(S As Complex3d)
```

```
S.Z=Near0(S.Z)
S.Y=Near0(S.Y)
S.X=Near0(S.X)
End Sub
Function Modulus(X As Double, Y As Double) As Double
Dim Resto As Double
Resto=X / Y - Fix(X / Y)
If Round(Resto * Y, MaxDigit)=0 Then
Modulus=0
Else
Modulus=Resto * Y
End If
End Function
Function ArcSin(X As Double) As Double
If Round(Abs(X), MaxDigit)=1 Then
ArcSin=Pi / 2 * Sgn(X)
Exit Function
End If
ArcSin=Atn(X / Sqr(1 - X ^ 2))
End Function
Function ArcCos(X As Double) As Double
If Round(Abs(X), MaxDigit)=1 Then
If X > 0 Then
ArcCos=0
Exit Function
Else
ArcCos=Pi
Exit Function
End If
End If
ArcCos=Atn(-X / Sqr(1 - X ^ 2)) + 2 * Atn(1)
End Function
Function Near0(X As Double) As Double
Dim R As Double
R=X
If Round_Results Then R=Round(R, MaxDigit)
Near0=R
If Abs(R) <= AsZero Then
Near0=0
End If
End Function
'----------------------------------
'END CORE 3d ALGEBRA
'----------------------------------
```

References

1. Walker MJ (1894) Quaternions as 4-Vectors. Am J Phys 24: 515.
2. Stephenson RJ (1966) Development of Vector Analysis from Quaternions. Am J Phys 34: 194.
3. Ilamed Y, Salingaros N (1981) Algebras with three anticommuting elements. I. Spinors and quaternions. J Math Phys 22: 2091.
4. Silva CC, de Andrade Martins R (2002) Polar and axial vectors versus quaternions. Am J Phys 70: 958.