```python
def basic_multivector_operations_3D():
    Print_Function()
    (ex,ey,ez) = MV.setup('e*x|y|z')
    A = MV('A','mv')
    A.Fmt(1,'A')
    A.Fmt(2,'A')
    A.Fmt(3,'A')
    A.even().Fmt(1,'%A_{+}')
    A.odd().Fmt(1,'%A_{-}')
    X = MV('X','vector')
    Y = MV('Y','vector')
    print 'g_{ij} = ',MV.metric
    X.Fmt(1,'X')
    Y.Fmt(1,'Y')
    (X*Y).Fmt(2,'X*Y')
    (X^Y).Fmt(2,'X^Y')
    (X|Y).Fmt(2,'X|Y')
    return
```

Code Output:

$$A = A + A^{x}e_{x} + A^{y}e_{y} + A^{z}e_{z} + A^{xy}e_{x}\wedge e_{y} + A^{xz}e_{x}\wedge e_{z} + A^{yz}e_{y}\wedge e_{z} + A^{xyz}e_{x}\wedge e_{y}\wedge e_{z}$$

$$\begin{aligned} A =& A \\ &+ A^{x}e_{x} + A^{y}e_{y} + A^{z}e_{z} \\ &+ A^{xy}e_{x}\wedge e_{y} + A^{xz}e_{x}\wedge e_{z} + A^{yz}e_{y}\wedge e_{z} \\ &+ A^{xyz}e_{x}\wedge e_{y}\wedge e_{z} \end{aligned}$$

$$\begin{aligned} A =& A \\ &+ A^{x}e_{x} \\ &+ A^{y}e_{y} \\ &+ A^{z}e_{z} \\ &+ A^{xy}e_{x}\wedge e_{y} \\ &+ A^{xz}e_{x}\wedge e_{z} \\ &+ A^{yz}e_{y}\wedge e_{z} \\ &+ A^{xyz}e_{x}\wedge e_{y}\wedge e_{z} \end{aligned}$$

$$g_{ij} = \begin{bmatrix} (e_{x}\cdot e_{x}) & (e_{x}\cdot e_{y}) & (e_{x}\cdot e_{z}) \\ (e_{x}\cdot e_{y}) & (e_{y}\cdot e_{y}) & (e_{y}\cdot e_{z}) \\ (e_{x}\cdot e_{z}) & (e_{y}\cdot e_{z}) & (e_{z}\cdot e_{z}) \end{bmatrix}$$

$$X = X^{x}e_{x} + X^{y}e_{y} + X^{z}e_{z}$$

$$Y = Y^{x}e_{x} + Y^{y}e_{y} + Y^{z}e_{z}$$

```python
def basic_multivector_operations_2D():
    Print_Function()
    (ex,ey) = MV.setup('e*x|y')
    print 'g_{ij} =',MV.metric
    X = MV('X','vector')
    A = MV('A','spinor')
    X.Fmt(1,'X')
    A.Fmt(1,'A')
    (X|A).Fmt(2,'X|A')
    (X<A).Fmt(2,'X<A')
    (A>X).Fmt(2,'A>X')
    return
```

Code Output:

$$g_{ij} = \begin{bmatrix} (e_x \cdot e_x) & (e_x \cdot e_y) \\ (e_x \cdot e_y) & (e_y \cdot e_y) \end{bmatrix}$$

$$X = X^x e_x + X^y e_y$$

$$A = A + A^{xy} e_x \wedge e_y$$

```
def basic_multivector_operations_2D_orthogonal():
    Print_Function()
    (ex,ey) = MV.setup('e*x|y',metric='[1,1]')
    print 'g_{ii} =',MV.metric
    X = MV('X','vector')
    A = MV('A','spinor')
    X.Fmt(1,'X')
    A.Fmt(1,'A')
    (X*A).Fmt(2,'X*A')
    (X|A).Fmt(2,'X|A')
    (X<A).Fmt(2,'X<A')
    (X>A).Fmt(2,'X>A')
    (A*X).Fmt(2,'A*X')
    (A|X).Fmt(2,'A|X')
    (A<X).Fmt(2,'A<X')
    (A>X).Fmt(2,'A>X')
    return
```

Code Output:

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

$$X = X^x e_x + X^y e_y$$

$$A = A + A^{xy} e_x \wedge e_y$$

```
def check_generalized_BAC_CAB_formulas():
    Print_Function()
    (a,b,c,d) = MV.setup('a b c d')
    print 'g_{ij} =',MV.metric
    print '\\bm{a|(b*c)} =',a|(b*c)
    print '\\bm{a|(b^c)} =',a|(b^c)
    print '\\bm{a|(b^c^d)} =',a|(b^c^d)
    print '\\bm{a|(b^c)+c|(a^b)+b|(c^a)} =',(a|(b^c))+(c|(a^b))+(b|(c^a))
    print '\\bm{a*(b^c)-b*(a^c)+c*(a^b)} =',a*(b^c)-b*(a^c)+c*(a^b)
    print '\\bm{a*(b^c^d)-b*(a^c^d)+c*(a^b^d)-d*(a^b^c)} =',a*(b^c^d)-b*(a^c^d)+c*(a^b^d)-d*(a^b^c)
    print '\\bm{(a^b)|(c^d)} =',(a^b)|(c^d)
    print '\\bm{((a^b)|c)|d} =',((a^b)|c)|d
    print '\\bm{(a^b)\\times (c^d)} =',Com(a^b,c^d)
    return
```

Code Output:

$$g_{ij} = \begin{bmatrix} (a \cdot a) & (a \cdot b) & (a \cdot c) & (a \cdot d) \\ (a \cdot b) & (b \cdot b) & (b \cdot c) & (b \cdot d) \\ (a \cdot c) & (b \cdot c) & (c \cdot c) & (c \cdot d) \\ (a \cdot d) & (b \cdot d) & (c \cdot d) & (d \cdot d) \end{bmatrix}$$

$$\boldsymbol{a \cdot (bc)} = -(a \cdot c) b + (a \cdot b) c$$

$$\boldsymbol{a \cdot (b \wedge c)} = -(a \cdot c) b + (a \cdot b) c$$

$$\boldsymbol{a \cdot (b \wedge c \wedge d)} = (a \cdot d) b \wedge c - (a \cdot c) b \wedge d + (a \cdot b) c \wedge d$$

$$\boldsymbol{a \cdot (b \wedge c) + c \cdot (a \wedge b) + b \cdot (c \wedge a)} = 0$$

$\boldsymbol{a(b\wedge c)-b(a\wedge c)+c(a\wedge b)} = 3a\wedge b\wedge c$

$\boldsymbol{a(b\wedge c\wedge d)-b(a\wedge c\wedge d)+c(a\wedge b\wedge d)-d(a\wedge b\wedge c)} = 4a\wedge b\wedge c\wedge d$

$\boldsymbol{(a\wedge b)\cdot(c\wedge d)} = -(a\cdot c)(b\cdot d)+(a\cdot d)(b\cdot c)$

$\boldsymbol{((a\wedge b)\cdot c)\cdot d} = -(a\cdot c)(b\cdot d)+(a\cdot d)(b\cdot c)$

$\boldsymbol{(a\wedge b)\times(c\wedge d)} = -(b\cdot d)\,a\wedge c+(b\cdot c)\,a\wedge d+(a\cdot d)\,b\wedge c-(a\cdot c)\,b\wedge d$

```
def rounding_numerical_components():
    Print_Function()
    (ex,ey,ez) = MV.setup('e_x e_y e_z',metric='[1,1,1]')
    X = 1.2*ex+2.34*ey+0.555*ez
    Y = 0.333*ex+4*ey+5.3*ez
    print 'X =',X
    print 'Nga(X,2) =',Nga(X,2)
    print 'X*Y =',X*Y
    print 'Nga(X*Y,2) =',Nga(X*Y,2)
    return
```

Code Output:

$X = 1\cdot 2e_x+2\cdot 34e_y+0\cdot 555e_z$

$Nga(X,2) = 1\cdot 2e_x+2\cdot 3e_y+0\cdot 55e_z$

$XY = 12\cdot 7011+4\cdot 02078e_x\wedge e_y+6\cdot 175185e_x\wedge e_z+10\cdot 182e_y\wedge e_z$

$Nga(XY,2) = 13\cdot 0+4\cdot 0e_x\wedge e_y+6\cdot 2e_x\wedge e_z+10\cdot 0e_y\wedge e_z$

```
def derivatives_in_rectangular_coordinates():
    Print_Function()
    X = (x,y,z) = symbols('x y z')
    (ex,ey,ez,grad) = MV.setup('e_x e_y e_z',metric='[1,1,1]',coords=X)
    f = MV('f','scalar',fct=True)
    A = MV('A','vector',fct=True)
    B = MV('B','grade2',fct=True)
    C = MV('C','mv')
    print 'f =',f
    print 'A =',A
    print 'B =',B
    print 'C =',C
    print 'grad*f =',grad*f
    print 'grad|A =',grad|A
    print 'grad*A =',grad*A
    print -MV.I
    print '-I*(grad^A) =',-MV.I*(grad^A)
    print 'grad*B =',grad*B
    print 'grad^B =',grad^B
    print 'grad|B =',grad|B
    return
```

Code Output:

$f = f$

$A = A^{x}e_x+A^{y}e_y+A^{z}e_z$

$B = B^{xy}e_x\wedge e_y+B^{xz}e_x\wedge e_z+B^{yz}e_y\wedge e_z$

$C = C+C^{x}e_x+C^{y}e_y+C^{z}e_z+C^{xy}e_x\wedge e_y+C^{xz}e_x\wedge e_z+C^{yz}e_y\wedge e_z+C^{xyz}e_x\wedge e_y\wedge e_z$

$\boldsymbol{\nabla} f = \partial_x f e_x+\partial_y f e_y+\partial_z f e_z$

$\boldsymbol{\nabla}\cdot A = \partial_x A^{x}+\partial_y A^{y}+\partial_z A^{z}$

$$\nabla A = (\partial_x A^x + \partial_y A^y + \partial_z A^z) + (-\partial_y A^x + \partial_x A^y)\, e_x \wedge e_y + (-\partial_z A^x + \partial_x A^z)\, e_x \wedge e_z + (-\partial_z A^y + \partial_y A^z)\, e_y \wedge e_z$$

$$-e_x \wedge e_y \wedge e_z$$

$$-I(\nabla \wedge A) = (-\partial_z A^y + \partial_y A^z)\, e_x + (\partial_z A^x - \partial_x A^z)\, e_y + (-\partial_y A^x + \partial_x A^y)\, e_z$$

$$\nabla B = (-\partial_y B^{xy} - \partial_z B^{xz})\, e_x + (\partial_x B^{xy} - \partial_z B^{yz})\, e_y + (\partial_x B^{xz} + \partial_y B^{yz})\, e_z + (\partial_z B^{xy} - \partial_y B^{xz} + \partial_x B^{yz})\, e_x \wedge e_y \wedge e_z$$

$$\nabla \wedge B = (\partial_z B^{xy} - \partial_y B^{xz} + \partial_x B^{yz})\, e_x \wedge e_y \wedge e_z$$

$$\nabla \cdot B = (-\partial_y B^{xy} - \partial_z B^{xz})\, e_x + (\partial_x B^{xy} - \partial_z B^{yz})\, e_y + (\partial_x B^{xz} + \partial_y B^{yz})\, e_z$$

```
def derivatives_in_spherical_coordinates():
    Print_Function()
    X = (r,th,phi) = symbols('r theta phi')
    curv = [[r*cos(phi)*sin(th),r*sin(phi)*sin(th),r*cos(th)],[1,r,r*sin(th)]]
    (er,eth,ephi,grad) = MV.setup('e_r e_theta e_phi',metric='[1,1,1]',coords=X,curv=curv)
    f = MV('f','scalar',fct=True)
    A = MV('A','vector',fct=True)
    B = MV('B','grade2',fct=True)
    print 'f =',f
    print 'A =',A
    print 'B =',B
    print 'grad*f =',grad*f
    print 'grad|A =',grad|A
    print '-I*(grad^A) =',(-MV.I*(grad^A)).simplify()
    print 'grad^B =',grad^B
```

Code Output:

$$f = f$$

$$A = A^r e_r + A^\theta e_\theta + A^\phi e_\phi$$

$$B = B^{r\theta} e_r \wedge e_\theta + B^{r\phi} e_r \wedge e_\phi + B^{\theta\phi} e_\theta \wedge e_\phi$$

$$\nabla f = \partial_r f e_r + \partial_\theta f e_\theta + \partial_\phi f e_\phi$$

$$\nabla \cdot A = \partial_\phi A^\phi + \partial_r A^r + \partial_\theta A^\theta$$

$$-I(\nabla \wedge A) = \left(\partial_\theta A^\phi - \partial_\phi A^\theta\right) e_r + \left(-\partial_r A^\phi + \partial_\phi A^r\right) e_\theta + \left(-\partial_\theta A^r + \partial_r A^\theta\right) e_\phi$$

$$\nabla \wedge B = \left(-\partial_\theta B^{r\phi} + \partial_\phi B^{r\theta} + \partial_r B^{\theta\phi}\right) e_r \wedge e_\theta \wedge e_\phi$$

```
def conformal_representations_of_circles_lines_spheres_and_planes():
    Print_Function()
    global n,nbar
    metric = '1 0 0 0 0,0 1 0 0 0,0 0 1 0 0,0 0 0 0 2,0 0 0 2 0'
    (e1,e2,e3,n,nbar) = MV.setup('e_1 e_2 e_3 n \\bar{n}',metric)
    print 'g_{ij} =',MV.metric
    e = n+nbar
    #conformal representation of points
    A = make_vector(e1)    # point a = (1,0,0)  A = F(a)
    B = make_vector(e2)    # point b = (0,1,0)  B = F(b)
    C = make_vector(-e1)   # point c = (-1,0,0) C = F(c)
    D = make_vector(e3)    # point d = (0,0,1)  D = F(d)
    X = make_vector('x',3)
    print 'F(a) =',A
    print 'F(b) =',B
    print 'F(c) =',C
    print 'F(d) =',D
    print 'F(x) =',X
    print '#a = e1, b = e2, c = -e1, and d = e3'
    print '#A = F(a) = 1/2*(a*a*n+2*a-nbar), etc.'
    print '#Circle through a, b, and c'
    print 'Circle: A^B^C^X = 0 =',(A^B^C^X)
```

```
    print '#Line through a and b'
    print 'Line   : A^B^n^X = 0 =',(A^B^n^X)
    print '#Sphere through a, b, c, and d'
    print 'Sphere: A^B^C^D^X = 0 =',(((A^B)^C)^D)^X
    print '#Plane through a, b, and d'
    print 'Plane : A^B^n^D^X = 0 =',(A^B^n^D^X)
    L = (A^B^e)^X
    L.Fmt(3,'Hyperbolic\\;\\; Circle: (A^B^e)^X = 0')
    return
```

Code Output:

$$g_{ij} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 2 & 0 \end{bmatrix}$$

$$F(a) = e_1 + \frac{1}{2}n - \frac{1}{2}\bar{n}$$

$$F(b) = e_2 + \frac{1}{2}n - \frac{1}{2}\bar{n}$$

$$F(c) = -e_1 + \frac{1}{2}n - \frac{1}{2}\bar{n}$$

$$F(d) = e_3 + \frac{1}{2}n - \frac{1}{2}\bar{n}$$

$$F(x) = x_1e_1 + x_2e_2 + x_3e_3 + \left(\frac{1}{2}(x_1)^2 + \frac{1}{2}(x_2)^2 + \frac{1}{2}(x_3)^2\right)n - \frac{1}{2}\bar{n}$$

a = e1, b = e2, c = -e1, and d = e3 A = F(a) = 1/2*(a*a*n+2*a-nbar), etc. Circle through a, b, and c

$$Circle : A \wedge B \wedge C \wedge X = 0 = -x_3e_1 \wedge e_2 \wedge e_3 \wedge n + x_3e_1 \wedge e_2 \wedge e_3 \wedge \bar{n} + \left(\frac{1}{2}(x_1)^2 + \frac{1}{2}(x_2)^2 + \frac{1}{2}(x_3)^2 - \frac{1}{2}\right)e_1 \wedge e_2 \wedge n \wedge \bar{n}$$

Line through a and b

$$Line : A \wedge B \wedge n \wedge X = 0 = -x_3e_1 \wedge e_2 \wedge e_3 \wedge n + \left(\frac{x_1}{2} + \frac{x_2}{2} - \frac{1}{2}\right)e_1 \wedge e_2 \wedge n \wedge \bar{n} + \frac{x_3}{2}e_1 \wedge e_3 \wedge n \wedge \bar{n} - \frac{x_3}{2}e_2 \wedge e_3 \wedge n \wedge \bar{n}$$

Sphere through a, b, c, and d

$$Sphere : A \wedge B \wedge C \wedge D \wedge X = 0 = \left(-\frac{1}{2}(x_1)^2 - \frac{1}{2}(x_2)^2 - \frac{1}{2}(x_3)^2 + \frac{1}{2}\right)e_1 \wedge e_2 \wedge e_3 \wedge n \wedge \bar{n}$$

Plane through a, b, and d

$$Plane : A \wedge B \wedge n \wedge D \wedge X = 0 = \left(-\frac{x_1}{2} - \frac{x_2}{2} - \frac{x_3}{2} + \frac{1}{2}\right)e_1 \wedge e_2 \wedge e_3 \wedge n \wedge \bar{n}$$

```
def properties_of_geometric_objects():
    global n,nbar
    Print_Function()
    metric = '# # # 0 0,'+ \
             '# # # 0 0,'+ \
             '# # # 0 0,'+ \
             '0 0 0 0 2,'+ \
             '0 0 0 2 0'
    (p1,p2,p3,n,nbar) = MV.setup('p1 p2 p3 n \\bar{n}',metric)
    print 'g_{ij} =',MV.metric
    P1 = F(p1)
    P2 = F(p2)
    P3 = F(p3)
```

```
    print '#%\\text{Extracting direction of line from }L = P1\\W P2\\W n'
    L = P1^P2^n
    delta = (L|n)|nbar
    print '(L|n)|\\bar{n} =',delta
    print '#%\\text{Extracting plane of circle from }C = P1\\W P2\\W P3'
    C = P1^P2^P3
    delta = ((C^n)|n)|nbar
    print '((C^n)|n)|\\bar{n}=',delta
    print '(p2-p1)^(p3-p1)=',(p2-p1)^(p3-p1)
    return
```

Code Output:

$$g_{ij} = \begin{bmatrix} (p_1 \cdot p_1) & (p_1 \cdot p_2) & (p_1 \cdot p_3) & 0 & 0 \\ (p_1 \cdot p_2) & (p_2 \cdot p_2) & (p_2 \cdot p_3) & 0 & 0 \\ (p_1 \cdot p_3) & (p_2 \cdot p_3) & (p_3 \cdot p_3) & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 2 & 0 \end{bmatrix}$$

Extracting direction of line from $L = P1 \wedge P2 \wedge n$

$(L \cdot n) \cdot \bar{n} = 2p_1 - 2p_2$

Extracting plane of circle from $C = P1 \wedge P2 \wedge P3$

$((C \wedge n) \cdot n) \cdot \bar{n} = 2p_1 \wedge p_2 - 2p_1 \wedge p_3 + 2p_2 \wedge p_3$

$(p2 - p1) \wedge (p3 - p1) = p_1 \wedge p_2 - p_1 \wedge p_3 + p_2 \wedge p_3$

```
def extracting_vectors_from_conformal_2_blade():
    Print_Function()
    print r'B = P1\W P2'
    metric = '0 -1 #,'+ \
             '-1 0 #,'+ \
             '# # #'
    (P1,P2,a) = MV.setup('P1 P2 a',metric)
    print 'g_{ij} =',MV.metric
    B = P1^P2
    Bsq = B*B
    print '%B^{2} =',Bsq
    ap = a-(a^B)*B
    print "a' = a-(a^B)*B =",ap
    Ap = ap+ap*B
    Am = ap-ap*B
    print "A+ = a'+a'*B =",Ap
    print "A- = a'-a'*B =",Am
    print '%(A+)^{2} =',Ap*Ap
    print '%(A-)^{2} =',Am*Am
    aB = a|B
    print 'a|B =',aB
    return
```

Code Output:

$B = P1 \wedge P2$

$$g_{ij} = \begin{bmatrix} 0 & -1 & (P_1 \cdot a) \\ -1 & 0 & (P_2 \cdot a) \\ (P_1 \cdot a) & (P_2 \cdot a) & (a \cdot a) \end{bmatrix}$$

$B^2 = 1$

$a' = a - (a \wedge B)B = -(P_2 \cdot a) P_1 - (P_1 \cdot a) P_2$

$A+ = a' + a'B = -2 (P_2 \cdot a) P_1$

$A- = a' - a'B = -2\left(P_1 \cdot a\right) P_2$

$(A+)^2 = 0$

$(A-)^2 = 0$

$a \cdot B = -\left(P_2 \cdot a\right) P_1 + \left(P_1 \cdot a\right) P_2$

```
def reciprocal_frame_test():
    Print_Function()
    metric = '1 # #,'+ \
             '# 1 #,'+ \
             '# # 1'
    (e1,e2,e3) = MV.setup('e1 e2 e3',metric)
    print 'g_{ij} =',MV.metric
    E = e1^e2^e3
    Esq = (E*E).scalar()
    print 'E =',E
    print '%E^{2} =',Esq
    Esq_inv = 1/Esq
    E1 = (e2^e3)*E
    E2 = (-1)*(e1^e3)*E
    E3 = (e1^e2)*E
    print 'E1 = (e2^e3)*E =',E1
    print 'E2 =-(e1^e3)*E =',E2
    print 'E3 = (e1^e2)*E =',E3
    w = (E1|e2)
    w = w.expand()
    print 'E1|e2 =',w
    w = (E1|e3)
    w = w.expand()
    print 'E1|e3 =',w
    w = (E2|e1)
    w = w.expand()
    print 'E2|e1 =',w
    w = (E2|e3)
    w = w.expand()
    print 'E2|e3 =',w
    w = (E3|e1)
    w = w.expand()
    print 'E3|e1 =',w
    w = (E3|e2)
    w = w.expand()
    print 'E3|e2 =',w
    w = (E1|e1)
    w = (w.expand()).scalar()
    Esq = expand(Esq)
    print '%(E1\\cdot e1)/E^{2} =',simplify(w/Esq)
    w = (E2|e2)
    w = (w.expand()).scalar()
    print '%(E2\\cdot e2)/E^{2} =',simplify(w/Esq)
    w = (E3|e3)
    w = (w.expand()).scalar()
    print '%(E3\\cdot e3)/E^{2} =',simplify(w/Esq)
    return
```

Code Output:

$$g_{ij} = \left[\begin{array}{ccc}1 & \left(e_{1}\cdot e_{2}\right) & \left(e_{1}\cdot e_{3}\right)\\\left(e_{1}\cdot e_{2}\right) & 1 & \left(e_{2}\cdot e_{3}\right)\\\left(e_{1}\cdot e_{3}\right) & \left(e_{2}\cdot e_{3}\right) & 1\end{array}\right]$$

$E = e_1 \wedge e_2 \wedge e_3$

$$E^2 = (e_1 \cdot e_2)^2 - 2\,(e_1 \cdot e_2)\,(e_1 \cdot e_3)\,(e_2 \cdot e_3) + (e_1 \cdot e_3)^2 + (e_2 \cdot e_3)^2 - 1$$

$$E1 = (e2 \wedge e3)E = \left((e_2 \cdot e_3)^2 - 1\right) e_1 + ((e_1 \cdot e_2) - (e_1 \cdot e_3)\,(e_2 \cdot e_3))\, e_2 + (-\,(e_1 \cdot e_2)\,(e_2 \cdot e_3) + (e_1 \cdot e_3))\, e_3$$

$$E2 = -(e1 \wedge e3)E = ((e_1 \cdot e_2) - (e_1 \cdot e_3)\,(e_2 \cdot e_3))\, e_1 + \left((e_1 \cdot e_3)^2 - 1\right) e_2 + (-\,(e_1 \cdot e_2)\,(e_1 \cdot e_3) + (e_2 \cdot e_3))\, e_3$$

$$E3 = (e1 \wedge e2)E = (-\,(e_1 \cdot e_2)\,(e_2 \cdot e_3) + (e_1 \cdot e_3))\, e_1 + (-\,(e_1 \cdot e_2)\,(e_1 \cdot e_3) + (e_2 \cdot e_3))\, e_2 + \left((e_1 \cdot e_2)^2 - 1\right) e_3$$

$$E1 \cdot e2 = 0$$

$$E1 \cdot e3 = 0$$

$$E2 \cdot e1 = 0$$

$$E2 \cdot e3 = 0$$

$$E3 \cdot e1 = 0$$

$$E3 \cdot e2 = 0$$

$$(E1 \cdot e1)/E^2 = 1$$

$$(E2 \cdot e2)/E^2 = 1$$

$$(E3 \cdot e3)/E^2 = 1$$