```
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:

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

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

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

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

$$A_{-} = A^{x}\boldsymbol{e_{x}} + A^{y}\boldsymbol{e_{y}} + A^{z}\boldsymbol{e_{z}} + A^{xyz}\boldsymbol{e_{x}}\wedge\boldsymbol{e_{y}}\wedge\boldsymbol{e_{z}}$$

$$g_{ij} = \left[\begin{smallmatrix} (e_x\cdot e_x) & (e_x\cdot e_y) & (e_x\cdot e_z) \\ (e_x\cdot e_y) & (e_y\cdot e_y) & (e_y\cdot e_z) \\ (e_x\cdot e_z) & (e_y\cdot e_z) & (e_z\cdot e_z) \end{smallmatrix}\right]$$

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

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

$$\begin{aligned} XY = & X^{x}Y^{x}\left(e_{x}\cdot e_{x}\right) + X^{x}Y^{y}\left(e_{x}\cdot e_{y}\right) + X^{x}Y^{z}\left(e_{x}\cdot e_{z}\right) + X^{y}Y^{x}\left(e_{x}\cdot e_{y}\right) + X^{y}Y^{y}\left(e_{y}\cdot e_{y}\right) + X^{y}Y^{z}\left(e_{y}\cdot e_{z}\right) + X^{z}Y^{x}\left(e_{x}\cdot e_{z}\right) + X^{z}Y^{y}\left(e_{y}\cdot e_{z}\right) + X^{z}Y^{z}\left(e_{z}\cdot e_{z}\right) \\ & + \left(X^{x}Y^{y} - X^{y}Y^{x}\right)\boldsymbol{e_{x}}\wedge\boldsymbol{e_{y}} + \left(X^{x}Y^{z} - X^{z}Y^{x}\right)\boldsymbol{e_{x}}\wedge\boldsymbol{e_{z}} + \left(X^{y}Y^{z} - X^{z}Y^{y}\right)\boldsymbol{e_{y}}\wedge\boldsymbol{e_{z}} \end{aligned}$$

$$X\wedge Y = \left(X^{x}Y^{y} - X^{y}Y^{x}\right)\boldsymbol{e_{x}}\wedge\boldsymbol{e_{y}} + \left(X^{x}Y^{z} - X^{z}Y^{x}\right)\boldsymbol{e_{x}}\wedge\boldsymbol{e_{z}} + \left(X^{y}Y^{z} - X^{z}Y^{y}\right)\boldsymbol{e_{y}}\wedge\boldsymbol{e_{z}}$$

$$X\cdot Y = X^{x}Y^{x}\left(e_{x}\cdot e_{x}\right) + X^{x}Y^{y}\left(e_{x}\cdot e_{y}\right) + X^{x}Y^{z}\left(e_{x}\cdot e_{z}\right) + X^{y}Y^{x}\left(e_{x}\cdot e_{y}\right) + X^{y}Y^{y}\left(e_{y}\cdot e_{y}\right) + X^{y}Y^{z}\left(e_{y}\cdot e_{z}\right) + X^{z}Y^{x}\left(e_{x}\cdot e_{z}\right) + X^{z}Y^{y}\left(e_{y}\cdot e_{z}\right) + X^{z}Y^{z}\left(e_{z}\cdot e_{z}\right)$$

```
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:

$$g_{ij} = \left[\begin{smallmatrix}(e_x\cdot e_x) & (e_x\cdot e_y)\\ (e_x\cdot e_y) & (e_y\cdot e_y)\end{smallmatrix}\right]$$

$$X = X^x \boldsymbol{e_x} + X^y \boldsymbol{e_y}$$

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

$$X \cdot A = \left(A^{xy} \left(- X^x \left(e_x\cdot e_y\right) - X^y \left(e_y\cdot e_y\right)\right)\right) \boldsymbol{e_x} + \left(A^{xy} \left(X^x \left(e_x\cdot e_x\right) + X^y \left(e_x\cdot e_y\right)\right)\right) \boldsymbol{e_y}$$

$$X \rfloor A = \left(A^{xy} \left(- X^x \left(e_x\cdot e_y\right) - X^y \left(e_y\cdot e_y\right)\right)\right) \boldsymbol{e_x} + \left(A^{xy} \left(X^x \left(e_x\cdot e_x\right) + X^y \left(e_x\cdot e_y\right)\right)\right) \boldsymbol{e_y}$$

$$A \lfloor X = \left(A^{xy} \left(X^x \left(e_x\cdot e_y\right) + X^y \left(e_y\cdot e_y\right)\right)\right) \boldsymbol{e_x} + \left(A^{xy} \left(- X^x \left(e_x\cdot e_x\right) - X^y \left(e_x\cdot e_y\right)\right)\right) \boldsymbol{e_y}$$

```
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:

$$g_{ii} = \left[\begin{smallmatrix}1\\ 1\end{smallmatrix}\right]$$

$$X = X^x \boldsymbol{e_x} + X^y \boldsymbol{e_y}$$

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

$$X A = \left(A X^x - A^{xy} X^y\right) \boldsymbol{e_x} + \left(A X^y + A^{xy} X^x\right) \boldsymbol{e_y}$$

$$X \cdot A = - A^{xy} X^y \boldsymbol{e_x} + A^{xy} X^x \boldsymbol{e_y}$$

$$X \rfloor A = - A^{xy} X^y \boldsymbol{e_x} + A^{xy} X^x \boldsymbol{e_y}$$

$$X \lfloor A = A X^x \boldsymbol{e_x} + A X^y \boldsymbol{e_y}$$

$$A X = \left(A X^x + A^{xy} X^y\right) \boldsymbol{e_x} + \left(A X^y - A^{xy} X^x\right) \boldsymbol{e_y}$$

$$A \cdot X = A^{xy} X^y \boldsymbol{e_x} - A^{xy} X^x \boldsymbol{e_y}$$

$$A \rfloor X = A X^x \boldsymbol{e_x} + A X^y \boldsymbol{e_y}$$

$$A \lfloor X = A^{xy} X^y \boldsymbol{e_x} - A^{xy} X^x \boldsymbol{e_y}$$

```
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:

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

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

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

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

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

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

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

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

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

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

```
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:

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

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

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

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

```
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 '-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:

$$f = f$$

$$A = A^{x}\boldsymbol{e_{x}} + A^{y}\boldsymbol{e_{y}} + A^{z}\boldsymbol{e_{z}}$$

$$B = B^{xy}\boldsymbol{e_{x}}\wedge \boldsymbol{e_{y}} + B^{xz}\boldsymbol{e_{x}}\wedge \boldsymbol{e_{z}} + B^{yz}\boldsymbol{e_{y}}\wedge \boldsymbol{e_{z}}$$

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

$$\boldsymbol{\nabla} f = \partial_{x} f\boldsymbol{e_{x}} + \partial_{y} f\boldsymbol{e_{y}} + \partial_{z} f\boldsymbol{e_{z}}$$

$$\boldsymbol{\nabla}\cdot A = \partial_{x} A^{x} + \partial_{y} A^{y} + \partial_{z} A^{z}$$

$$\boldsymbol{\nabla} A = \partial_{x} A^{x} + \partial_{y} A^{y} + \partial_{z} A^{z} + \left(- \partial_{y} A^{x} + \partial_{x} A^{y}\right)\boldsymbol{e_{x}}\wedge \boldsymbol{e_{y}} + \left(- \partial_{z} A^{x} + \partial_{x} A^{z}\right)\boldsymbol{e_{x}}\wedge \boldsymbol{e_{z}} + \left(- \partial_{z} A^{y} + \partial_{y} A^{z}\right)\boldsymbol{e_{y}}\wedge \boldsymbol{e_{z}}$$

$$-I(\boldsymbol{\nabla}\wedge A) = \left(- \partial_{z} A^{y} + \partial_{y} A^{z}\right)\boldsymbol{e_{x}} + \left(\partial_{z} A^{x} - \partial_{x} A^{z}\right)\boldsymbol{e_{y}} + \left(- \partial_{y} A^{x} + \partial_{x} A^{y}\right)\boldsymbol{e_{z}}$$

$$\boldsymbol{\nabla} B = \left(- \partial_{y} B^{xy} - \partial_{z} B^{xz}\right)\boldsymbol{e_{x}} + \left(\partial_{x} B^{xy} - \partial_{z} B^{yz}\right)\boldsymbol{e_{y}} + \left(\partial_{x} B^{xz} + \partial_{y} B^{yz}\right)\boldsymbol{e_{z}} + \left(\partial_{z} B^{xy} - \partial_{y} B^{xz} + \partial_{x} B^{yz}\right)\boldsymbol{e_{x}}\wedge \boldsymbol{e_{y}}\wedge \boldsymbol{e_{z}}$$

$$\boldsymbol{\nabla}\wedge B = \left(\partial_{z} B^{xy} - \partial_{y} B^{xz} + \partial_{x} B^{yz}\right)\boldsymbol{e_{x}}\wedge \boldsymbol{e_{y}}\wedge \boldsymbol{e_{z}}$$

$$\boldsymbol{\nabla}\cdot B = \left(- \partial_{y} B^{xy} - \partial_{z} B^{xz}\right)\boldsymbol{e_{x}} + \left(\partial_{x} B^{xy} - \partial_{z} B^{yz}\right)\boldsymbol{e_{y}} + \left(\partial_{x} B^{xz} + \partial_{y} B^{yz}\right)\boldsymbol{e_{z}}$$

```
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:

$$f = f$$

$$A = A^{r}\boldsymbol{e_{r}} + A^{\theta}\boldsymbol{e_{\theta}} + A^{\phi}\boldsymbol{e_{\phi}}$$

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

$$\boldsymbol{\nabla} f = \partial_{r} f\boldsymbol{e_{r}} + \frac{\partial_{\theta} f}{r}\boldsymbol{e_{\theta}} + \frac{\partial_{\phi} f}{r \sin{\left(\theta \right)}}\boldsymbol{e_{\phi}}$$

$$\boldsymbol{\nabla}\cdot A = \partial_{r} A^{r} + \frac{A^{\theta}}{r \tan{\left(\theta \right)}} + 2 \frac{A^{r}}{r} + \frac{\partial_{\theta} A^{\theta}}{r} + \frac{\partial_{\phi} A^{\phi}}{r \sin{\left(\theta \right)}}$$

$$-I(\boldsymbol{\nabla}\wedge A) = \left(\frac{A^{\phi}\cos(\theta) + \sin(\theta)\partial_{\theta}A^{\phi} - \partial_{\phi}A^{\theta}}{r\sin(\theta)}\right)\boldsymbol{e_r} + \left(-\partial_r A^{\phi} - \frac{A^{\phi}}{r} + \frac{\partial_{\phi}A^{r}}{r\sin(\theta)}\right)\boldsymbol{e_\theta} + \left(\frac{r\partial_r A^{\theta} + A^{\theta} - \partial_{\theta}A^{r}}{r}\right)\boldsymbol{e_\phi}$$

$$\boldsymbol{\nabla}\wedge B = \left(\partial_r B^{\theta\phi} + 2\frac{B^{\theta\phi}}{r} - \frac{B^{r\phi}}{r\tan(\theta)} - \frac{\partial_{\theta}B^{r\phi}}{r} + \frac{\partial_{\phi}B^{r\theta}}{r\sin(\theta)}\right)\boldsymbol{e_r}\wedge\boldsymbol{e_\theta}\wedge\boldsymbol{e_\phi}$$

```
def noneuclidian_distance_calculation():
    Print_Function()
    from sympy import solve,sqrt
    metric = '0 # #,# 0 #,# # 1'
    (X,Y,e) = MV.setup('X Y e',metric)
    print 'g_{ij} =',MV.metric
    print '%(X\\W Y)^{2} =',(X^Y)*(X^Y)
    L = X^Y^e
    B = L*e # D&L 10.152
    Bsq = (B*B).scalar()
    print '#%L = X\\W Y\\W e \\text{ is a non-euclidian line}'
    print 'B = L*e =',B
    BeBr =B*e*B.rev()
    print '%BeB^{\\dagger} =',BeBr
    print '%B^{2} =',B*B
    print '%L^{2} =',L*L # D&L 10.153
    (s,c,Binv,M,S,C,alpha,XdotY,Xdote,Ydote) = symbols('s c (1/B) M S C alpha (X.Y) (X.e) (Y.e)')
    Bhat = Binv*B # D&L 10.154
    R = c+s*Bhat # Rotor R = exp(alpha*Bhat/2)
    print '#%s = \\f{\\sinh}{\\alpha/2} \\text{ and } c = \\f{\\cosh}{\\alpha/2}'
    print '%e^{\\alpha B/{2\\abs{B}}} =',R
    Z = R*X*R.rev() # D&L 10.155
    Z.obj = expand(Z.obj)
    Z.obj = Z.obj.collect([Binv,s,c,XdotY])
    Z.Fmt(3,'%RXR^{\\dagger}')
    W = Z|Y # Extract scalar part of multivector
    # From this point forward all calculations are with sympy scalars
    #print '#Objective is to determine value of C = cosh(alpha) such that W = 0'
    W = W.scalar()
    print '%W = Z\\cdot Y =',W
    W = expand(W)
    W = simplify(W)
    W = W.collect([s*Binv])
    M = 1/Bsq
    W = W.subs(Binv**2,M)
    W = simplify(W)
    Bmag = sqrt(XdotY**2-2*XdotY*Xdote*Ydote)
    W = W.collect([Binv*c*s,XdotY])
    #Double angle substitutions
    W = W.subs(2*XdotY**2-4*XdotY*Xdote*Ydote,2/(Binv**2))
    W = W.subs(2*c*s,S)
    W = W.subs(c**2,(C+1)/2)
    W = W.subs(s**2,(C-1)/2)
    W = simplify(W)
    W = W.subs(1/Binv,Bmag)
    W = expand(W)
    print '#%S = \\f{\\sinh}{\\alpha} \\text{ and } C = \\f{\\cosh}{\\alpha}'
    print 'W =',W
    Wd = collect(W,[C,S],exact=True,evaluate=False)
    Wd_1 = Wd[ONE]
    Wd_C = Wd[C]
    Wd_S = Wd[S]
    print '%\\text{Scalar Coefficient} =',Wd_1
    print '%\\text{Cosh Coefficient} =',Wd_C
    print '%\\text{Sinh Coefficient} =',Wd_S
```

```
    print '%\\abs{B} =',Bmag
    Wd_1 = Wd_1.subs(Bmag,1/Binv)
    Wd_C = Wd_C.subs(Bmag,1/Binv)
    Wd_S = Wd_S.subs(Bmag,1/Binv)
    lhs = Wd_1+Wd_C*C
    rhs = -Wd_S*S
    lhs = lhs**2
    rhs = rhs**2
    W = expand(lhs-rhs)
    W = expand(W.subs(1/Binv**2,Bmag**2))
    W = expand(W.subs(S**2,C**2-1))
    W = W.collect([C,C**2],evaluate=False)
    a = simplify(W[C**2])
    b = simplify(W[C])
    c = simplify(W[ONE])
    print '#%\\text{Require } aC^{2}+bC+c = 0'
    print 'a =',a
    print 'b =',b
    print 'c =',c
    x = Symbol('x')
    C =  solve(a*x**2+b*x+c,x)[0]
    print '%b^{2}-4ac =',simplify(b**2-4*a*c)
    print '%\\f{\\cosh}{\\alpha} = C = -b/(2a) =',expand(simplify(expand(C)))
    return
```

Code Output:

$$g_{ij} = \begin{bmatrix} 0 & (X\cdot Y) & (X\cdot e) \\ (X\cdot Y) & 0 & (Y\cdot e) \\ (X\cdot e) & (Y\cdot e) & 1 \end{bmatrix}$$

$(X \wedge Y)^2 = (X \cdot Y)^2$

$L = X \wedge Y \wedge e$ is a non-euclidian line

$B = Le = \boldsymbol{X} \wedge \boldsymbol{Y} - (Y \cdot e)\, \boldsymbol{X} \wedge \boldsymbol{e} + (X \cdot e)\, \boldsymbol{Y} \wedge \boldsymbol{e}$

$BeB^{\dagger} = ((X \cdot Y)(-(X \cdot Y) + 2\,(X \cdot e)(Y \cdot e)))\, \boldsymbol{e}$

$B^2 = (X \cdot Y)((X \cdot Y) - 2\,(X \cdot e)(Y \cdot e))$

$L^2 = (X \cdot Y)((X \cdot Y) - 2\,(X \cdot e)(Y \cdot e))$

$s = \sinh(\alpha/2)$ and $c = \cosh(\alpha/2)$

$e^{\alpha B/2|B|} = c + (1/B)s\boldsymbol{X} \wedge \boldsymbol{Y} - (1/B)(Y \cdot e)\, s\boldsymbol{X} \wedge \boldsymbol{e} + (1/B)(X \cdot e)\, s\boldsymbol{Y} \wedge \boldsymbol{e}$

$$\begin{aligned} RXR^{\dagger} = &\left((1/B)^2 (X \cdot Y)^2 s^2 - 2(1/B)^2 (X \cdot Y)(X \cdot e)(Y \cdot e)\, s^2 + 2(1/B)(X \cdot Y)\, cs - 2(1/B)(X \cdot e)(Y \cdot e)\, cs + c^2\right) \boldsymbol{X} \\ &+ 2(1/B)(X \cdot e)^2\, cs\boldsymbol{Y} \\ &+ (2(1/B)(X \cdot Y)(X \cdot e)\, s\,(-(1/B)(X \cdot Y)\, s + 2(1/B)(X \cdot e)(Y \cdot e)\, s - c))\, \boldsymbol{e} \end{aligned}$$

$W = Z \cdot Y = (1/B)^2 (X \cdot Y)^3 s^2 - 4(1/B)^2 (X \cdot Y)^2 (X \cdot e)(Y \cdot e)\, s^2 + 4(1/B)^2 (X \cdot Y)(X \cdot e)^2 (Y \cdot e)^2 s^2 + 2(1/B)(X \cdot Y)^2 cs - 4(1/B)(X \cdot Y)(X \cdot e)(Y \cdot e)\, cs + (X \cdot Y)\, c^2$

$S = \sinh(\alpha)$ and $C = \cosh(\alpha)$

$W = C\,(X \cdot Y) - C\,(X \cdot e)(Y \cdot e) + S\sqrt{(X \cdot Y)^2 - 2\,(X \cdot Y)(X \cdot e)(Y \cdot e)} + (X \cdot e)(Y \cdot e)$

Scalar Coefficient $= (X \cdot e)(Y \cdot e)$

Cosh Coefficient $= (X \cdot Y) - (X \cdot e)(Y \cdot e)$

Sinh Coefficient $= \sqrt{(X \cdot Y)^2 - 2\,(X \cdot Y)(X \cdot e)(Y \cdot e)}$

$|B| = \sqrt{(X \cdot Y)^2 - 2\,(X \cdot Y)(X \cdot e)(Y \cdot e)}$

Require $aC^2 + bC + c = 0$

$a = (X \cdot e)^2 (Y \cdot e)^2$

$b = 2 (X \cdot e) (Y \cdot e) ((X \cdot Y) - (X \cdot e) (Y \cdot e))$

$c = (X \cdot Y)^2 - 2 (X \cdot Y) (X \cdot e) (Y \cdot e) + (X \cdot e)^2 (Y \cdot e)^2$

$b^2 - 4ac = 0$

$\cosh(\alpha) = C = -b/(2a) = -\frac{(X \cdot Y)}{(X \cdot e) (Y \cdot e)} + 1$

```
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:

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

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

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

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

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

$$F(x) = x_1\boldsymbol{e_1} + x_2\boldsymbol{e_2} + x_3\boldsymbol{e_3} + \left(\frac{1}{2}(x_1)^2 + \frac{1}{2}(x_2)^2 + \frac{1}{2}(x_3)^2\right)\boldsymbol{n} - \frac{1}{2}\boldsymbol{\bar{n}}$$

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

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

Line through a and b

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

Sphere through a, b, c, and d

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

Plane through a, b, and d

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

$$\begin{aligned} Hyperbolic\ Circle : (A \wedge B \wedge e) \wedge X = 0 = & - x_3 \boldsymbol{e_1} \wedge \boldsymbol{e_2} \wedge \boldsymbol{e_3} \wedge \boldsymbol{n} \\ & - x_3 \boldsymbol{e_1} \wedge \boldsymbol{e_2} \wedge \boldsymbol{e_3} \wedge \boldsymbol{\bar{n}} \\ & + \left(-\frac{1}{2}(x_1)^2 + x_1 - \frac{1}{2}(x_2)^2 + x_2 - \frac{1}{2}(x_3)^2 - \frac{1}{2}\right) \boldsymbol{e_1} \wedge \boldsymbol{e_2} \wedge \boldsymbol{n} \wedge \boldsymbol{\bar{n}} \\ & + x_3 \boldsymbol{e_1} \wedge \boldsymbol{e_3} \wedge \boldsymbol{n} \wedge \boldsymbol{\bar{n}} \\ & - x_3 \boldsymbol{e_2} \wedge \boldsymbol{e_3} \wedge \boldsymbol{n} \wedge \boldsymbol{\bar{n}} \end{aligned}$$

```
def properties_of_geometric_objects():
    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 $L = P1 \wedge P2 \wedge n$

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

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

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

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

```
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:

$B = P1 \wedge P2$

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

$B^2 = 1$

$a' = a - (a \wedge B)B = -(P_2 \cdot a)\,\boldsymbol{P_1} - (P_1 \cdot a)\,\boldsymbol{P_2}$

$A+ = a' + a'B = -2\,(P_2 \cdot a)\,\boldsymbol{P_1}$

$A- = a' - a'B = -2\,(P_1 \cdot a)\,\boldsymbol{P_2}$

$(A+)^2 = 0$

$(A-)^2 = 0$

$a \cdot B = -(P_2 \cdot a)\,\boldsymbol{P_1} + (P_1 \cdot a)\,\boldsymbol{P_2}$

```
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:

$$g_{ij} = \left[\begin{smallmatrix} 1 & (e_1\cdot e_2) & (e_1\cdot e_3) \\ (e_1\cdot e_2) & 1 & (e_2\cdot e_3) \\ (e_1\cdot e_3) & (e_2\cdot e_3) & 1 \end{smallmatrix}\right]$$

$$E = \boldsymbol{e_1} \wedge \boldsymbol{e_2} \wedge \boldsymbol{e_3}$$

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

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

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

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

$$E1 \cdot e2 = 0$$

$$E1 \cdot e3 = 0$$

$$E2 \cdot e1 = 0$$

$$E2 \cdot e3 = 0$$

$$E3 \cdot e1 = 0$$

$$E3 \cdot e2 = 0$$

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

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

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