BRAID VARIETIES (V)

2026/01/??

1. Differential forms on G

The materials are from [1, Section 3,5] [2, Section 3] and [3, Section 9].

We will embed $G \hookrightarrow GL_n$. In the computation below, most of computation reduces to the case GL_n. We are working over the ring

$$\Omega_G = \mathbb{C}[x_{ij}]_{\text{loc}}\langle dx_{ij}\rangle$$

which is the algebra over $\mathbb{C}[x_{ij}]_{\text{loc}}$ generated by anti-commutative variables dx_{ij}. Sometimes we write $\wedge$ to emphasize that we are working with a non-commutative ring. Let

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1n} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nn} \end{bmatrix}$$

be a generic element of GL_n.

Let us consider the **matrix-valued 1-form on G**

$$\theta_L(X) = X^{-1} \cdot dX \in M_n(\Omega^1_G), \qquad \theta_R(X) = dX \cdot X^{-1} \in M_n(\Omega^1_G).$$

The forms are left-invariant and right-invariant respectively. That is, if we change variable $X \mapsto AX$, then

$$\theta_L(X) = X^{-1}dX \mapsto \theta_L(AX) = (AX)^{-1}d(AX) = X^{-1}A^{-1}AdX = X^{-1}dX = \theta_L(X).$$

Define a **bi-invariant 3-form on G** by

$$\Omega(X) = \frac{1}{3}\operatorname{Tr}(\theta_L(X)^3) = \frac{1}{3}\operatorname{Tr}(\theta_R(X)^3) \in \Omega^3_G.$$

Note: $\operatorname{Tr}(AB) = (-1)^{|A||B|}\operatorname{Tr}(BA)$ where $|A|, |B|$ are the differential degrees. One might prefer the following more Lie-theoretic form of Ω:

$$\Omega = \frac{1}{6}\kappa\big([\theta_L, \theta_L], \theta_L\big), \qquad \kappa(-,-) = \operatorname{Tr}(- \cdot -).$$

Note that after base change to the ring of differential forms, the usual Lie bracket $[,]$ becomes the super commutator, i.e. $[A,B] = AB - (-1)^{|A||B|}BA$.

Define a **2-form on $G \times G$**

$$(X|Y) = \operatorname{Tr}\left(\theta_L(X)\theta_R(Y)\right) \in \Omega^2_{G\times G}$$

where $X \times Y$ is a generic element in $G \times G$. We can check the following relations

$$d(X,Y) = \Omega(X) + \Omega(Y) - \Omega(XY)$$
$$0 = (Y,Z) - (XY,Z) + (X,YZ) - (X,Y) \qquad \text{(cocyle condition)}.$$

Let me briefly explain the motivation of defining these differential forms.

- It is well-known that the Lie algebra of G can be identified with the space of right-invariant vector fields. In our case $G = GL_n$, the entry of $dX \cdot X^{-1}$ forms a dual basis of the standard basis of $\mathfrak{gl}_n$. Similar for left-invariant vector fields, but the geometric and algebraic brackets differ by a sign.
- One can prove the space of bi-invariant k-forms over G coincides with the de Rham cohomology $H^k_{dR}(G;\mathbb{C})$. When G is simple,
$$\dim H^k_{dR}(G;\mathbb{C}) = 1,0,0,1,\ldots \qquad k = 0,1,2,3,\ldots.$$
The form Ω is the canonical generator of $H^3_{dR}(G;\mathbb{C})$.
- As a generator of minimal degree, it is easy to see the cohomology class $[\Omega]$ is primitive,
$$\mathsf{mult}^*[\Omega] = [\Omega] \otimes 1 + 1 \otimes [\Omega], \text{ i.e. } [\Omega(XY)] = [\Omega(X)] + [\Omega(Y)].$$
However, this is not true before taking cohomology class. The difference is a differential of a 2-form over $G \times G$. The 2-form $(X|Y)$ is one of the choice.
- Associativity of mult implies
$$[\Omega(XYZ)] = [\Omega(X)] + [\Omega(Y)] + [\Omega(Z)].$$
There are two ways of proving it via $d\big((Y,Z) - (X,YZ)\big)$ and $d\big((X,Y) - (XY,Z)\big)$. They are the same before taking differential d.

Example. When $G = \mathbb{C}^\times$.

$$\theta_R(z) = \theta_L(x) = \frac{dx}{x} = d\log(x), \quad \Omega(x) = 0, \quad (x|y) = \frac{dx \wedge dy}{xy} = d\log(x) \wedge d\log(y).$$

Example. Assume $G = GL_2$. The right-invariant form

$$\begin{aligned}\theta_R(X) = dX \cdot X^{-1} &= \begin{bmatrix} dx_{11} & dx_{12} \\ dx_{21} & dx_{22} \end{bmatrix} \begin{bmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{bmatrix}^{-1} \\ &= \frac{1}{x_{11}x_{22} - x_{12}x_{21}} \begin{bmatrix} dx_{11} & dx_{12} \\ dx_{21} & dx_{22} \end{bmatrix} \begin{bmatrix} x_{22} & -x_{12} \\ -x_{21} & x_{11} \end{bmatrix} \\ &= \frac{1}{x_{11}x_{22} - x_{12}x_{21}} \begin{bmatrix} x_{22}dx_{11} - x_{21}dx_{12} & -x_{12}dx_{11} + x_{11}dx_{12} \\ x_{22}dx_{21} - x_{21}dx_{22} & -x_{12}dx_{21} + x_{11}dx_{22} \end{bmatrix}.\end{aligned}$$

The 3-form $\Omega(X)$ is given by

$$\frac{1}{(x_{11}x_{22} - x_{12}x_{21})^2} \begin{pmatrix} x_{22}dx_{11} \wedge dx_{12} \wedge dx_{21} - x_{21}dx_{11} \wedge dx_{12} \wedge dx_{22} \\ + x_{12}dx_{11} \wedge dx_{21} \wedge dx_{22} - x_{11}dx_{12} \wedge dx_{21} \wedge dx_{22} \end{pmatrix}.$$

The 2-form $(X|Y)$ is

$$\frac{1}{(x_{11}x_{12}-x_{12}x_{21})(y_{11}y_{12}-y_{12}y_{21})}\begin{pmatrix} (x_{21}y_{12}+x_{22}y_{22})\,dx_{11}\wedge dy_{11} \\ -(x_{11}y_{12}+x_{12}y_{22})\,dx_{21}\wedge dy_{11} \\ -(x_{21}y_{11}+x_{22}y_{21})\,dx_{11}\wedge dy_{12} \\ +(x_{11}y_{11}+x_{12}y_{21})\,dx_{21}\wedge dy_{12} \\ +(x_{21}y_{12}+x_{22}y_{22})\,dx_{12}\wedge dy_{21} \\ -(x_{11}y_{12}+x_{12}y_{22})\,dx_{22}\wedge dy_{21} \\ -(x_{21}y_{11}+x_{22}y_{21})\,dx_{12}\wedge dy_{22} \\ +(x_{11}y_{11}+x_{12}y_{21})\,dx_{22}\wedge dy_{22} \end{pmatrix}.$$

A **twisted symplectic variety** is pair (f,ω) where

$$\text{a smooth } M \xrightarrow{f} G \text{ and } \omega\in\Omega^2_M \text{ such that } d\omega=-f^*\Omega.$$

The most important construction is, we can construct product. We define $(f_1,\omega_1)\times(f_2,\omega_2)=(f,\omega)$ by

$$f: M_1\times M_2\to G\times G\overset{\mathsf{mult}}{\to} G,\qquad \omega=\omega_1+\omega_2+(f_1|f_2).$$

This is well-defined since

$$d\omega=d\omega_1+d\omega_2+d(f_1|f_2)=-\Omega(f_1)-\Omega(f_2)+(\Omega(f_1)+\Omega(f_2)-\Omega(f_1f_2)).$$

The cocycle condition implies the associativity

$$\big((f_1,\omega_1)\times(f_2,\omega_2)\big)\times(f_3,\omega_3)=(f_1,\omega_1)\times\big((f_2,\omega_2)\times(f_3,\omega_3)\big).$$

To describe the product of more factors we introduce a 2-form over $G^{\times\ell}$

$$(X_1|X_2|\dots|X_\ell)\quad\text{with}\quad(\cdots|X|Y|\cdots)=(\cdots|XY|\cdots)+(X|Y).$$

This is well-defined by cocycle condition. Assume $(f_1,\omega_1)\times\cdots\times(f_\ell,\omega_\ell)=(f,\omega)$, we have

$$\begin{aligned} f:&M_1\times\cdots\times M_\ell\to G\times\cdots\times G\overset{\mathsf{mult}}{\to} G,\\ \omega&=\omega_1+\cdots+\omega_\ell+(f_1|\cdots|f_\ell).\end{aligned}$$

Example. When $G=\mathbb{C}^\times$. It is easy to compute

$$(x_1|\cdots|x_n)=\sum_{1\le i<j\le n}\frac{dx_idx_j}{x_ix_j}.$$

2. Differential forms over braid varieties

2.1. **A 2-form.** Recall

$$X(\beta)=\{(z_k): B_{i_1}(z_1)\cdots B_{i_\ell}(z_\ell)\in uB\},\qquad u=\text{Demazure product of }\beta.$$

We define a 2-form over $X(\beta,u)$ by

$$\omega=(B_{i_1}(z_1)|\cdots|B_{i_2}(z_2)).$$

Note that there are relations between $z_1,\dots,z_\ell$.

Example. Consider $G = SL_2$. Note that

$$\theta_L(B_1(z)) = \begin{bmatrix} 0 & \\ -dz & 0 \end{bmatrix}, \qquad \theta_R(B_1(z)) = \begin{bmatrix} 0 & dz \\ & 0 \end{bmatrix}.$$

For $\beta = \sigma_1^2$, we have

$$\omega = \mathrm{Trace}\left(\begin{bmatrix} 0 & \\ -dz_1 & 0 \end{bmatrix}\begin{bmatrix} 0 & dz_2 \\ & 0 \end{bmatrix}\right) = -dz_1 \wedge dz_2 = 0.$$

The last equality follows from the relation $z_1 z_2 = 1$.

For $\beta = \sigma_1^3$, we have

$$\omega = -dz_1 \wedge dz_2 + (-z_2^2) dz_1 \wedge dz_3 - dz_2 \wedge dz_3 = 2\frac{dz_2 \wedge dz_3}{z_2 z_3 - 1}.$$

The last relation follows from $z_1 z_2 z_3 = z_1 + z_3$. Obvious that

$$\omega = 2\frac{dz_2 \wedge d(z_2 z_3 - 1)}{z_2(z_2 z_3 - 1)} = -2\frac{dz_3 \wedge d(z_2 z_3 - 1)}{z_3(z_2 z_3 - 1)}.$$

Note that $z_2, z_3, \boxed{z_2 z_3 - 1}$ are cluster variables.

Theorem. For any Demazure weave $\mathfrak{W} : \beta \to \sigma_{w_0}$, using the $\{\mathsf{A}_v\}$-variable,

$$\omega \in \mathrm{span}_{\mathbb{Z}}\left(\frac{d\mathsf{A}_{v_1} d\mathsf{A}_{v_2}}{\mathsf{A}_{v_1}\mathsf{A}_{v_2}}\right).$$

Note that the coefficients are functions in $\{\mathsf{A}_v\}$ a prior .

Proof. Let us consider the waive variety $X(\mathfrak{W}^{=})$. For each region walk from the leftmost region to the rightmost region, we can define a 2-form, by substituting $B_i(\tilde{\mathsf{z}}_e, u_e)$ of $\tilde{\mathsf{U}}$ in the $(\cdots|\cdots|\cdots)$.

(1) We need to show the 2-form on the bottom walk satisfying the condition;
(2) We need to show when cross a vertex, the difference satisfying the condition.

Notice that

$$u, u' \text{ are Laurent polynomials in } \{\mathsf{A}_v\} \Longrightarrow \frac{du du'}{uu'} \in \mathrm{span}_{\mathbb{Z}}\left(\frac{d\mathsf{A}_{v_1} d\mathsf{A}_{v_2}}{\mathsf{A}_{v_1}\mathsf{A}_{v_2}}\right).$$

The claim (2) follows from direct computation (see Appendix). Let us prove claim (1). On the bottom walk, the 2-form looks like

$$(B_{i_1}(0, u_1)|\cdots|B_{i_l}(0, u_l)|\tilde{\mathsf{U}}_1|\tilde{\mathsf{U}}_2|\ldots).$$

Note that $B_i(0, u) = B_i(0)\chi_i(u)$. Since $B_i(0) = \dot{s}_i$ is a constant, we have

$$(\cdots|\dot{s}_i|\cdots) = (\cdots\dot{s}_i|\cdots) = (\cdots|\dot{s}_i\cdots)$$
$$(\dot{s}_i|X|\cdots) = (\dot{s}_i X|\cdots) = (X|\cdots).$$

So we have

$$(\cdots|u^\gamma|\dot{s}_i|\cdots) = (\cdots|u^\gamma\dot{s}_i|\cdots) = (\cdots|\dot{s}_i u^{s_i\gamma}|\cdots) = (\cdots|\dot{s}_i|u^{s_i\gamma}|\cdots).$$

Then we can equivalently written it as

$$(u_1^{\gamma_1} \mid \cdots \mid u_l^{\gamma_l} \mid \tilde{\mathsf{U}}_1 \mid \tilde{\mathsf{U}}_2 \mid \cdots) = (u_1^{\gamma_1} \mid \cdots \mid u_l^{\gamma_l}) + (u_1^{\gamma_1} \cdots u_l^{\gamma_l} \mid \tilde{\mathsf{U}}_1 \mid \tilde{\mathsf{U}}_2 \mid \cdots).$$

Note that

$$(u_1^{\gamma_1} \mid \cdots \mid u_l^{\gamma_l}) \in \operatorname{span}_{\mathbb{Z}}\left(\frac{d\mathsf{A}_{v_1} d\mathsf{A}_{v_2}}{\mathsf{A}_{v_1}\mathsf{A}_{v_2}}\right)$$

by the computation for $G = \mathbb{C}^\times$. Note that

$$X \in B, Y \in \operatorname{Rad}(B) \Longrightarrow (X|Y) = 0 \qquad \text{(since the diagonal entries of } dY \text{ vanish).}$$

The second term vanishes. □

2.2. **Quivers.** Let us construct an **ice quiver**. The vertices are trivalent vectices $\mathfrak{W}_3$. The frozen vertices are those $v \in \mathfrak{W}_3$ appearing in some bottom u-labeling. We define the quiver

$$\#\{v_1 \to v_2\} \text{ or } -\#\{v_1 \leftarrow v_2\} = \text{coefficient of } \frac{d\mathsf{A}_{v_1} d\mathsf{A}_{v_2}}{\mathsf{A}_{v_1}\mathsf{A}_{v_2}} \text{ in } \frac{1}{2}\omega.$$

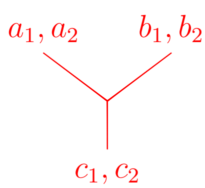
where at least one of v_1, v_2 is unfrozen. We will see the number of arrows is an integer. From the proof, there is a combinatorial description of the coefficients of ω and thus the quiver. It has three types of contributions

- the bottom boundary;
- the trivalent vertex;
- the hexavalent vertex.

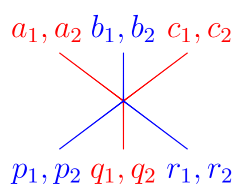
For an unfrozen vertex v_1 and an arbitrary vertex v_2, the bottom boundary has no contribution. Let us abbreviate $\mathsf{A}_{v_1} = \mathsf{A}_1$ and $\mathsf{A}_{v_2} = \mathsf{A}_2$. Note that

$$\begin{matrix} u = \mathsf{A}_1^{a_1}\mathsf{A}_2^{a_2}\cdots \\ u' = \mathsf{A}_1^{a_1'}\mathsf{A}_2^{a_2'}\cdots \end{matrix} \Longrightarrow \frac{du\,du'}{uu'} = \begin{vmatrix} a_1 & a_2 \\ a_1' & a_2' \end{vmatrix} \frac{d\mathsf{A}_1 d\mathsf{A}_2}{\mathsf{A}_1\mathsf{A}_2} + \cdots.$$

Then the contribution of trivalent vertices and hexavalent vertices (called **intersection number**)



$$\begin{vmatrix} 1 & 1 & 1 \\ a_1 & c_1 & b_1 \\ a_2 & c_2 & b_2 \end{vmatrix},$$



$$\frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \end{vmatrix} - \frac{1}{2}\begin{vmatrix} 1 & 1 & 1 \\ p_1 & q_1 & r_1 \\ p_2 & q_2 & r_2 \end{vmatrix}.$$

In the above diagram, the labeling is the exponents of A_1 and A_2. Note that $p_i + q_i = b_i + c_i$ and $q_i + r_i = a_i + b_i$, we have

$$\begin{vmatrix} 1 & 1 & 1 \\ a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \end{vmatrix} = \begin{vmatrix} 1 & 2 & 2 \\ a_1 & a_1 + b_1 & b_1 + c_1 \\ a_1 & a_2 + b_2 & b_2 + c_2 \end{vmatrix} \equiv \begin{vmatrix} q_1 + r_1 & p_1 + q_1 \\ q_2 + r_2 & p_2 + q_2 \end{vmatrix} \mod 2,$$

$$\begin{vmatrix} 1 & 1 & 1 \\ p_1 & q_1 & r_1 \\ p_2 & q_2 & r_2 \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 \\ p_1 & p_1+q_1 & q_1+r_1 \\ p_2 & p_2+q_2 & q_2+r_2 \end{vmatrix} \equiv \begin{vmatrix} p_1+q_1 & q_1+r_1 \\ p_2+q_2 & q_2+r_2 \end{vmatrix} \mod 2.$$

The description of quiver is equivalent to the pure combinatorial description in [3, Section 4.4, 4.5, 4.6].

References

[1] Anton Mellit. Toric stratifications of character varieties *Publications mathématiques de l'IHÉS* arXiv:1905.10685. (old name: Cell decompositions of character varieties) 1

[2] Roger Casals, Eugene Gorsky, Mikhail Gorsky, José Simental. Algebraic weaves and braid varieties. *American Journal of Mathematics*, arXiv:2012.06931. 1

[3] Roger Casals, Eugene Gorsky, Mikhail Gorsky, Ian Le, Linhui Shen, José Simental. Cluster structures on braid varieties. *Journal of the American Mathematical Society*, 2025. arXiv:2207.11607. 1, 2.2

Appendix A. Computation

Firstly, we need to define the class of matrices over non-commutative rings which is not properly defined in SageMath.

```
class Matrix(list): # for any ring
    def __add__(self,other):
        res = [[self[i][j]+other[i][j] for j in range(len(other[0]))] for i in range(len(self))]
        return Matrix(res)
    def __sub__(self,other):
        res = [[self[i][j]-other[i][j] for j in range(len(other[0]))] for i in range(len(self))]
        return Matrix(res)
    def __mul__(self,other):
        if type(other)==Matrix:
            res = [[sum(self[i][k]*other[k][j] for k in range(len(self[0]))) for j in range(len(other[0]))] for i in range(len(self))]
            return Matrix(res)
        else:
            return Matrix([[other*self[i][j] for j in range(len(self[0]))] for i in range(len(self))])
    def __pow__(self,n):
        if n==0:
            return Matrix([[(1 if i==j else 0) for j in range(len(self[0]))] for i in range(len(self))])
        elif n==-1:
            res = matrix(self)^(-1)
            return Matrix([list(row) for row in res])
        elif n < -1:
            return (self^(-1))^(-n)
        else:
            return self^(n-1)*self
    def trace(self):
        return sum(self[i][i] for i in range(len(self)))
    def __repr__(self):
        res = ""
        width = [max(len(str(self[i][j])) for i in range(len(self))) for j in range(len(self[0]))]
        for i in range(len(self)):
            row = "  ".join(str(self[i][j]).rjust(width[j]) for j in range(len(self[0])))
            res+= "["+row+"]\n"
        return res[:-1]

def d(omega):
    if type(omega) == type(Diff.an_element()):
        mydict= omega.dict()
        def monomial(*arg):
```

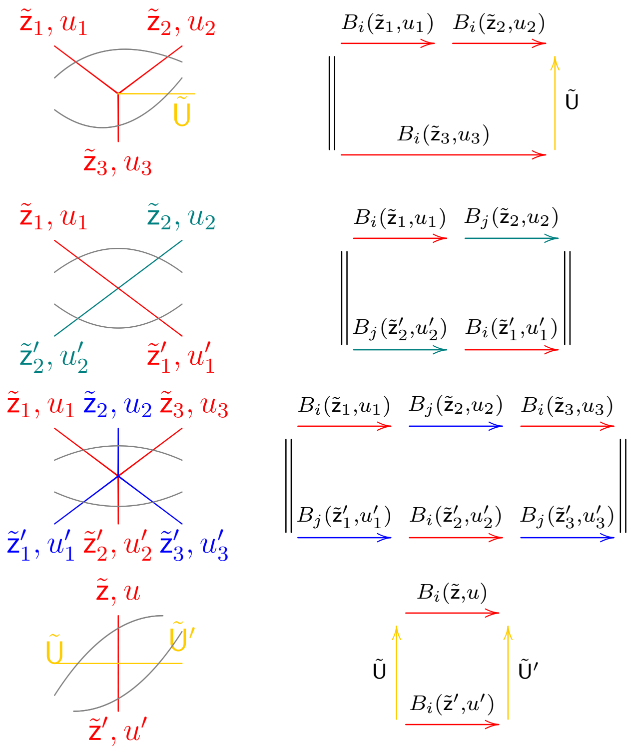
```
            res = Diff(1)
            for i in range(len(Diff.gens())):
                if arg[i]==1:
                    res = res*Diff.gens()[i]
            return res
        return sum(d(mydict[ind])*monomial(*ind) for ind in mydict)
    if type(omega) == Matrix:
        return Matrix([[d(omega[i][j]) for j in range(len(omega[0]))] for i in range(
    len(omega))])
    elif "derivative" not in dir(omega):
        return 0
    else:
        return sum(omega.derivative(Poly.gens()[i])*Diff.gens()[i] for i in range(len(
    Poly.gens())))
def thetaL(M):
    return M^(-1)*d(M)
def thetaR(M):
    return d(M)*M^(-1)
def beta(*ary):
    if len(ary)<=1: return 0
    if len(ary)==2: return (thetaL(ary[0])*thetaR(ary[1])).trace()
    newary = list(ary[:-2])+[ary[-2]*ary[-1]]
    return beta(*newary)+beta(ary[-2],ary[-1])
def Omega(M):
    return 1/3*(thetaL(M)^3).trace()
```

```
n = 2
x_names = ["x"+str(i)+str(j) for i in [1..n] for j in [1..n]]
x_names+= ["y"+str(i)+str(j) for i in [1..n] for j in [1..n]]
x_names+= ["z"+str(i)+str(j) for i in [1..n] for j in [1..n]]

d_names = ["d"+var for var in x_names]
Poly = PolynomialRing(QQ,x_names).fraction_field(); exec("\n".join("%s = Poly.gens()[%s
    ]"%(x_names[i],i) for i in range(len(x_names))))
Diff = GradedCommutativeAlgebra(Poly,d_names);

X = Matrix([[Poly.gens()[i*n+j] for j in range(n)]for i in range(n)])
Y = Matrix([[Poly.gens()[n^2+i*n+j] for j in range(n)]for i in range(n)])
Z = Matrix([[Poly.gens()[2*n^2+i*n+j] for j in range(n)]for i in range(n)])
print(thetaL(X))
print(beta(X,Y))
print(Omega(X) == 1/3*(thetaR(X)^3).trace())
print(d(beta(X,Y)) == Omega(X)+Omega(Y)-Omega(X*Y))
print(0 == beta(Y,Z)-beta(X*Y,Z)+beta(X,Y*Z)-beta(X,Y))
```

Appendix B. Explicit Computation



$$\begin{aligned}&\omega - \omega'\\ &= -2\frac{du_1du_2}{u_1u_2} + 2\frac{du_1du_3}{u_1u_3} - 2\frac{du_2du_3}{u_2u_3}\end{aligned}$$

$$\omega = \omega' = 0$$

$$\omega = \frac{du_1du_2}{u_1u_2} - \frac{du_1du_3}{u_1u_3} + \frac{du_2du_3}{u_2u_3}$$
$$\omega' = \frac{du'_1du'_2}{u'_1u'_2} - \frac{du'_1du'_3}{u'_1u'_3} + \frac{du'_2du'_3}{u'_2u'_3}$$

$$\omega = \omega' = 0$$

```
x_names = ["u1","u2","u3","z3"]

d_names = ["d"+var for var in x_names]
Poly = PolynomialRing(QQ,x_names).fraction_field(); exec("\n".join("%s = Poly.gens()[%s
    ]"%(x_names[i],i) for i in range(len(x_names))))
Diff = GradedCommutativeAlgebra(Poly,d_names);

z2 = u3/(u1*u2)
z1 = z3 + u2/(u1*u3)
a = -u1/(u2*u3)
B = lambda z,u: Matrix([[u*z,-u^(-1)],[u,0]])
U = Matrix([[1,a],[0,1]])
print(B(z1,u1)*B(z2,u2)==B(z3,u3)*U)
beta(B(z1,u1),B(z2,u2)) - beta(B(z3,u3),U)
```

```
True
-2/(u1*u2)*du1*du2 + 2/(u1*u3)*du1*du3 - 2/(u2*u3)*du2*du3
```

```
x_names = ["u1","u2","z1","z2"]

d_names = ["d"+var for var in x_names]
```

```
Poly = PolynomialRing(QQ,x_names).fraction_field(); exec("\n".join("%s = Poly.gens()[%s
    ]"%(x_names[i],i) for i in range(len(x_names))))
Diff = GradedCommutativeAlgebra(Poly,d_names);

B1 = lambda z,u: Matrix([[u*z,-u^(-1),0,0],[u,0,0,0],[0,0,1,0],[0,0,0,1]])
B3 = lambda z,u: Matrix([[1,0,0,0],[0,1,0,0],[0,0,u*z,-u^(-1)],[0,0,u,0]])
beta(B1(z1,u1),B3(z2,u2))
```

```
0
```

```
x_names = ["u1","u2","u3","z1","z2","z3"]

d_names = ["d"+var for var in x_names]
Poly = PolynomialRing(QQ,x_names).fraction_field(); exec("\n".join("%s = Poly.gens()[%s
    ]"%(x_names[i],i) for i in range(len(x_names))))
Diff = GradedCommutativeAlgebra(Poly,d_names);

B1 = lambda z,u: Matrix([[u*z,-u^(-1),0],[u,0,0],[0,0,1]])
B2 = lambda z,u: Matrix([[1,0,0],[0,u*z,-u^(-1)],[0,u,0]])

beta(B1(z1,u1),B2(z2,u2),B1(z3,u3))
```

```
1/(u1*u2)*du1*du2 - 1/(u1*u3)*du1*du3 + 1/(u2*u3)*du2*du3
```

```
x_names = ["u","z","zp","a"]

d_names = ["d"+var for var in x_names]
Poly = PolynomialRing(QQ,x_names).fraction_field(); exec("\n".join("%s = Poly.gens()[%s
    ]"%(x_names[i],i) for i in range(len(x_names))))
Diff = GradedCommutativeAlgebra(Poly,d_names);

ap = 0; up = u; a = -z+zp

B = lambda z,u: Matrix([[u*z,-u^(-1)],[u,0]])
U = Matrix([[1,a],[0,1]]); Up = Matrix([[1,ap],[0,1]])
print(U*B(z,u)==B(zp,up)*Up)
print(beta(U,B(z,u)))
print(beta(B(zp,up),Up))
```

```
True
0
0
```