

N=14_Algebraic_proof

July 20, 2021

MULTIPLE EXTREMAL DISC-PACKINGS IN COMPACT HYPERBOLIC SURFACES

** CASE N = 14: ALGEBRAIC PROOF**

An NEC-group uniformizing a non-orientable surface of genus p=6 that contains at least two extremal 3-packings Summary of the particular data for this file. It all comes from the numerical experiments:

- N=14 in 1)
 - A particular set of side-pairing transformations in 6)
 - The two bananas: side-pairing 2, banana 2, sign -1 and side-pairing 10, banana 1, sign -1 in 8)
 - The point z~0.516-0.248i is the first intersection point in 9) (indexed 0, since sagemath first index is 0)
 - The point w in 11) is not the center of the disc. The exponent j equals 2.
 - A particular set of products of side-pairings in 14)
-

Recall that $u = e^R$ where $\cosh R = \frac{1}{2 \sin \frac{\pi}{N}}$.

Therefore $u + u^{-1} = \frac{u^2 + 1}{u} = \frac{1}{\sin \frac{\pi}{N}}$, thus $\sin \frac{\pi}{N} u^2 - u + \sin \frac{\pi}{N} = 0$ and we have

$$u = \frac{1 + \sqrt{1 - 4 \sin^2 \frac{\pi}{N}}}{2 \sin \frac{\pi}{N}}.$$

```
[19]: #Approximate value of u
N=14
print (e^(arccosh(1/(2*sin(pi/N))))).n(), (( 1+sqrt( 1-4*(sin(pi/N))^2 ) )/
→(2*sin(pi/N))).n()
```

4.25917177632979 4.25917177632979

1) We compute the minimal polynomial of u , assigned to the variable pol .

[20] :

```
N=14  
pol=(( 1+sqrt( 1-4*(sin(pi/N))^2 ) )/(2*sin(pi/N))).minpoly()  
show(pol)
```

```
x^6 - 4*x^5 - x^4 - x^2 - 4*x + 1
```

[21] : #Approximate value of the roots of pol: notice that u is the largest real root
show(pol.roots(ring=QQbar))

```
[(0.2347874311051429?, 1),  
(4.259171776329791?, 1),  
(-0.8019377358048382? - 0.5974076228942927?*I, 1),  
(-0.8019377358048382? + 0.5974076228942927?*I, 1),  
(0.5549581320873712? - 0.8318782793354422?*I, 1),  
(0.5549581320873712? + 0.8318782793354422?*I, 1)]
```

2) We denote the field $\mathbb{Q}(u)$ as K1.

[22] : K1.<u> = NumberField(pol,embedding=3)

CC(u)

#The instruction CC gives the approximate value of an element of the field K1.

#The given embedding into the reals is chosen so that CC(u) gives the right value

[22] : 4.25917177632979

3) Since $u = e^{\frac{\pi i}{N}}$, we see that $\sin \frac{\pi}{N}$ is the field element given by $\frac{u}{u^2 + 1}$. We call it sn.

[23] : print(CC(u/(u^2+1)),sin(pi/N).n()) #A numerical verification

sn=u/(u^2+1)

show(sn) #Displays sn as an element of K1

```
(0.222520933956320, 0.222520933956314)
```

```
1/8*u^4 - 1/2*u^3 - 1/4*u^2 + 1/2*u + 1/8
```

4) Accordingly, $\cos^2 \frac{\pi}{N} = 1 - \sin^2 \frac{\pi}{N} = 1 - \frac{u^2}{(u^2+1)^2}$ is the field element given by $\frac{u^4+u^2+1}{(u^2+1)^2}$.

We call $v = \cos \frac{\pi}{N}$. It is an element of $\mathbb{Q}(u)$ if and only if $u^4 + u^2 + 1$ is a square in K1. We also denote it as cs as a nickname.

We denote $L = \mathbb{Q}(u, v)$. This field agrees with K1 when $v \in K1$, of course.

[24] : `print(CC(sqrt(u^4+u^2+1)/(u^2+1)),(cos(pi/N)).n()) #Numerical verification`

```

if (u^4+u^2+1).is_square():
    v=sqrt(u^4+u^2+1)/(u^2+1) # cos(pi/N)
    print(CC(v))
    cs=v
    show(v) #Displays v as an element of K1
    L=K1
else:
    Ring.<t> = PolynomialRing(K1)
    L.<v> = K1.extension(t^2-(u^4+u^2+1)/((u^2+1)^2))
    cs=v

#The code below this line constructs the righ embedding of L into the complex
#numbers
#that matches the numerical values we know that u and v must have.
#We call this embedding L_approx
#It will be very important later, in particular in points 8) and 11)
j=0
test=0
while test!=1:
    L_approx = L.embeddings(CC)[j]
    if abs(CC(sqrt(u^4+u^2+1)/(u^2+1))-L_approx(v))<10^-3 and
       abs(CC(u)-L_approx(u))<10^(-3): test+=1
    j+=1
print L_approx(u),L_approx(v) #This line must give the values we knew in advance
#(numerical verification)

```

(0.974927912181824, 0.974927912181824)

4.25917177632979 0.974927912181839

5) The matrices of a, b, c are

$$a = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \quad b = \begin{pmatrix} 0 & u^{-1} \\ u & 0 \end{pmatrix}, \quad c = \begin{pmatrix} -\cos \frac{\pi}{N} & \sin \frac{\pi}{N} \\ \sin \frac{\pi}{N} & \cos \frac{\pi}{N} \end{pmatrix}$$

[25] : `a=matrix(2,[1,0,0,-1])
b=matrix(2,[0,u^(-1),u,0])
c=matrix(2,[-cs,sn,sn,cs])
show(a)
show(b)
show(c)`

[1 0]
[0 -1]

```

[          0 -u^5 + 4*u^4 + u^3 + u + 4]
[          u                               0]
[          -v 1/8*u^4 - 1/2*u^3 - 1/4*u^2 + 1/2*u + 1/
8]
[1/8*u^4 - 1/2*u^3 - 1/4*u^2 + 1/2*u + 1/8
v]

```

6) We introduce in the list s the side-pairings for this surface group: our conjecture, driven by numerical experiments, is that the surface group thus generated will contain more than one extremal packing.

We use the notation $G_i = (ca)^i(abcb)(ca)^{-i}$.

```
[26]: def G(i):
    return (c*a)^i*(a*b*c*b)*(c*a)^(-i)

s=['zero'] #for s[j] to denote the j-th side-pairing we add an "empty" item in position 0
s.append((c*a)^(-4)*b*(c*a)) #s1
s.append(G(0)^7*(c*a)^(-2)) #s2
s.append((c*a)^(-3)*a*G(0)^(-1)) #s3
s.append(G(0)^9*G(1)^2) #s4
s.append((c*a)^7*(a*b)*(c*a)^2) #s5
s.append(G(0)^2*a*(c*a)^6) #s6
s.append(G(0)^3*(c*a)^9) #s7
s.append((c*a)^6*(b)*(c*a)^5) #s8
s.append(G(1)^5*(c*a)^(-3)) #s9
s.append(G(0)^(-4)*(c*a)^(-3)) #s10
s.append(G(1)^5*G(0)^4) #s11
s.append(G(1)^7*(c*a)*(a*b)*(c*a)^(-1)*G(1)^2) #s12
s.append(G(0)^(-3)*a*b*G(0)^(-6)) #s13
s.append(G(1)^3*G(0)^(-5)) #s14
s.append(G(1)^(-3)*(c*a)*a*b*(c*a)^(-1)*G(1)^(-3)) #s15
s.append(G(1)^(-5)*G(0)^(-6)) #s16
s.append(G(1)^(-6)*G(0)) #s17
s.append(G(1)^(-6)*(c*a)*a*b*(c*a)^(-1)*G(1)^(-2)) #s18

show(s[1:]) # How do the matrices look like? Their coefficients lie in L, but are they too horrible?
```

[
[-1/4*u^5 + u^4 + 1/4*u^3 + 1/4*u^2 + 3/4 (1/4*u^5 - 3/4*u^4 - u^3 - u^2
↳ 5/4*u - 9/4)*v] [(-2*u^5 + 31/4*u^4 + 3*u^3 + 1/2*u^2 + u + 31/4)*v
↳ -7/2*u^5 + 105/8*u^4 + 27/4*u^3 + 7/4*u^2 + 17/4*u + 121/8] [1/4*u^5
↳ u^4 - 1/4*u^3 - 1/4*u^2 - 3/4 (-1/4*u^5 + 3/4*u^4 + u^3 + u^2 + 5/4*u + 9/
↳ 4)*v] [(2*u^5 - 31/4*u^4 - 3*u^3 - 1/2*u^2 - u - 31/4)*v 7/2*u^5 - 105/
↳ 8*u^4 - 27/4*u^3 - 7/4*u^2 - 17/4*u - 121/8] [-1/8*u^5 + 1/2*u^4 + 1/4*u^3 -
↳ 1/4*u^2 - 1/8*u + 1/4 (1/4*u^5 - u^4 - 1/2*u^3 + u^2 + 1/4*u)*v]
↳ [-15/8*u^5 + 57/8*u^4 + 13/4*u^3 + u^2 + 13/8*u + 59/8 (1/4*u^5 - u^4 -
↳ 1/2*u^3 + u^2 + 1/4*u - 1)*v] [-19/8*u^5 + 71/8*u^4 + 19/4*u^3 + 5/4*u^2 + 17/
↳ 8*u + 79/8 (-u^5 + 7/2*u^4 + 3*u^3 + 1/2*u^2 + u + 4)*v] [
↳ -1/4*u^5 + 7/8*u^4 + 3/4*u^3 + 1/2*u + 5/8 (1/4*u^5 - 5/4*u^4 + 1/2*u^3 +
↳ u^2 + 1/4*u - 3/4)*v] [23/8*u^5 - 43/4*u^4 - 23/4*u^3 - 5/4*u^2 - 21/8*u - 12
↳ (9/4*u^5 - 17/2*u^4 - 4*u^3 - 3/2*u^2 - 13/4*u - 9)*v] [(-2*u^5 + 15/2*u^4 +
↳ 4*u^3 + 1/2*u^2 + 2*u + 8)*v -5/2*u^5 + 75/8*u^4 + 5*u^3 + 3/4*u^2 + 2*u + 87/
↳ 8] [(-1/4*u^5 + 3/4*u^4 + u^3 + 1/2*u^2 + 1/4*u + 7/4)*v -115/8*u^5 +
↳ 54*u^4 + 55/2*u^3 + 27/4*u^2 + 131/8*u + 245/4] [5/2*u^5 - 19/2*u^4 - 17/
↳ 4*u^3 - 5/4*u^2 - 9/4*u - 41/4 (29/4*u^5 - 109/4*u^4 - 14*u^3 - 3*u^2 - 29/4*u
↳ - 119/4)*v] [-1/4*u^5 + u^4 + 1/4*u^3 + 1/4*u^2 + 3/4 (41/
↳ 4*u^5 - 155/4*u^4 - 19*u^3 - 3*u^2 - 45/4*u - 177/4)*v] [-1/
↳ 2*u^5 + 15/8*u^4 + u^3 + 3/4*u^2 + 15/8 (11*u^5 - 83/2*u^4 - 41/2*u^3 - 9/2*u^2
↳ - 23/2*u - 46)*v] [(3/4*u^5 - 5/2*u^4 - 5/2*u^3 - u^2 - 5/4*u - 7/2)*v 23/
↳ 4*u^5 - 173/8*u^4 - 11*u^3 - 5/2*u^2 - 23/4*u - 195/8] [(1/2*u^5 -
↳ 7/4*u^4 - 3/2*u^3 - u^2 - u - 5/4)*v -89/8*u^5 + 167/4*u^4 + 43/2*u^3 + 5*u^2 +
↳ 101/8*u + 191/4] [(3/4*u^5 - 11/4*u^4 - 3/2*u^3 - u^2 - 1/4*u - 13/4)*v 27/
↳ 4*u^5 - 203/8*u^4 - 51/4*u^3 - 7/2*u^2 - 8*u - 229/8] [(3/4*u^5 - 11/4*u^4 -
↳ 3/2*u^3 - u^2 - 1/4*u - 13/4)*v 27/4*u^5 - 203/8*u^4 - 51/4*u^3 - 7/2*u^2 - 8*u
↳ - 229/8]

```

[(-1/4*u^5 + 3/4*u^4 + u^3 + u^2 + 5/4*u + 1/4)*v   3/4*u^5 - 11/4*u^4 - 7/4*u^3 - u
-1/4*u^2 - u - 3], [           -1/8*u^4 + 1/4*u^3 + 1/4*u^2 + 1/4*u - 1/8
→ (-1/4*u^4 + u^3 + 1/2*u^2 + u - 1/4)*v], [ (1/4*u^5 - 3/4*u^4
→ - u^3 - u^2 - 5/4*u - 1/4)*v -3/4*u^5 + 11/4*u^4 + 7/4*u^3 + 1/4*u^2 + u + 3], 
→ [           1/8*u^4 - 1/4*u^3 - 1/4*u^2 - 1/4*u + 1/8
→ (1/4*u^4 - u^3 - 1/2*u^2 - u + 1/4)*v], [ (1/4*u^5 - u^4 - 1/2*u^3 + u^2
+ 1/4*u - 2)*v -7/8*u^5 + 13/4*u^4 + 7/4*u^3 + 3/4*u^2 + 9/8*u + 7/2], [
→ (-1/4*u^5 + u^4 + 1/2*u^3 - u^2 - 1/4*u + 1)*v           1/8*u^5 - 3/8*u^4
→ -3/4*u^3 - 3/8*u - 1/8], [           (-1/2*u^2 + u + 1/
-2)*v -1/8*u^5 + 3/8*u^4 + 3/4*u^3 + 1/4*u^2 - 1/8*u + 3/8], [(3/4*u^5 - 11/
→ 4*u^4 - 3/2*u^3 - u^2 - 9/4*u - 13/4)*v           -1/4*u^5 + 7/8*u^4 + 3/4*u^3
+ 1/2*u + 13/8], [ (-3/4*u^5 + 3*u^4 + u^3 - 1/2*u^2 + 3/4*u + 5/2)*v -3/
→ 8*u^5 + 3/2*u^4 + 1/4*u^3 - 1/4*u^2 + 5/8*u + 7/4], [           1/8*u^4
→ 1/2*u^3 - 3/4*u^2 - 3/8           (1/2*u^2 - 1/2)*v], [
→ 1/8*u^5 - 3/4*u^4 + 1/4*u^2 + 3/8*u - 1/2           (13/4*u^5 - 49/
→ 4*u^4 - 6*u^3 - 3/2*u^2 - 17/4*u - 53/4)*v], [ (-3/4*u^5 + 11/4*u^4 +
→ 2*u^3 + u^2 + 3/4*u + 9/4)*v           -3/2*u^5 + 23/4*u^4 + 9/4*u^3 + 3/4*u^2 + 5/
→ 4*u + 13/2], [ (1/4*u^5 - 3/4*u^4 - u^3 + u^2 + 3/4*u - 1/4)*v
→ -3/4*u^5 + 11/4*u^4 + 7/4*u^3 + 1/4*u^2 + u + 3], [ (1/
→ 2*u^3 + 1/2*u^2 + 1/2*u - 1/2)*v           -2*u^5 + 59/8*u^4 + 9/2*u^3 + 3/4*u^2 +
→ 2*u + 67/8], [ -1/4*u^5 + 7/8*u^4 + u^3 + 1/2*u^2 + 1/4*u + 9/8 (-7/
→ 4*u^5 + 13/2*u^4 + 7/2*u^3 + u^2 + 13/4*u + 15/2)*v], [           -1/8*u^5
+ 1/4*u^4 + 1/2*u^3 + 5/8*u + 1/4           (7/2*u^5 - 53/4*u^4 - 13/2*u^3 - u^2
→ 3*u - 55/4)*v], [ -1/4*u^5 + 9/8*u^4 + 1/4*u^3 - 1/2*u^2 + 7/8 (-7/
→ 4*u^5 + 27/4*u^4 + 5/2*u^3 + u^2 + 9/4*u + 29/4)*v], [           -1/4*u^5 + 9/
→ 8*u^4 + 1/4*u^3 - 1/2*u^2 + 7/8 (-7/4*u^5 + 27/4*u^4 + 5/2*u^3 + u^2 + 9/4*u +
→ 29/4)*v]
]

```

7) We compute $\cosh^2 \frac{d_n}{2}$ for the distances d_0, d_1, d_2 associated to a tiling of hyperbolic space with regular N-gons of angle $2\pi/3$. We store the three values in the list ch.

First, as $d_0 = 2R$, then $\cosh^2 \frac{d_0}{2} = \cosh R = \frac{(u^2+1)^2}{4u^2}$.

Second, $\cosh d_1 = \cosh^2(d_0) - \sinh^2(d_0) \cos(4\pi/N) = (1 - \cos(4\pi/N)) \cosh^2(d_0) + \cos(4\pi/N)$.

Now, since $\cosh d = 2 \cosh^2 \frac{d}{2} - 1$, we have

$$\cosh^2 \left(\frac{d_1}{2} \right) = \frac{1 + \cosh d_1}{2} = \frac{1 + (1 - \cos(4\pi/N)) \cosh^2(d_0) + \cos(4\pi/N)}{2}$$

that is

$$\cosh^2\left(\frac{d_1}{2}\right) = \frac{1 + \cosh d_1}{2} = \frac{1 + (1 - \cos(4\pi/N))(2 \cosh^2 \frac{d_0}{2} - 1)^2 + \cos(4\pi/N)}{2}.$$

Accordingly, we have $\cosh d_2 = \cosh^2(d_0) - \sinh^2(d_0) \cos(6\pi/N) = (1 - \cos(6\pi/N)) \cosh^2(d_0) + \cos(6\pi/N)$ and therefore

$$\cosh^2\left(\frac{d_2}{2}\right) = \frac{1 + \cosh d_2}{2} = \frac{1 + (1 - \cos(6\pi/N))(2 \cosh^2 \frac{d_0}{2} - 1)^2 + \cos(6\pi/N)}{2}.$$

NOTE: Recall that $\cos \frac{k\pi}{N}$ is the value at $x = \cos \frac{\pi}{N}$ of the k -th Chebyshev polynomial.

```
[27]: ch=[(u^2+1)^2/(4*u^2)]
ch.append((1+(1-chebyshev_T(4,cs))*(2*ch[0]-1)^2+chebyshev_T(4,cs))/(2))
ch.append((1+(1-chebyshev_T(6,cs))*(2*ch[0]-1)^2+chebyshev_T(6,cs))/(2))
```

```
show(ch) # How do these values look like? They lie in L, but are they too
         →horrible?
```

```
[-u^5 + 15/4*u^4 + 2*u^3 + 1/2*u^2 + u + 19/4,
 -7/2*u^5 + 53/4*u^4 + 13/2*u^3 + 3/2*u^2 + 4*u + 61/4,
 -7*u^5 + 53/2*u^4 + 13*u^3 + 3*u^2 + 8*u + 61/2]
```

8) We compute the equations of the two relevant bananas, and the coordinates of the intersection. We need to recall which bananas are the promising ones in each case (meaning for which side-pairing and which distance d_j) from the information we obtained numerically.

The two variables *sign* can be (independently) 1 or -1: we guess the right choice after computing (via CC) the numerical value of the intersection, that we expect to match what we obtained in our numerical experiments.

```
[28]: def H(k,s): #A function that occurs in the equation of the banana
    if s.det() == -1:
        return 4*ch[k]
    elif s.det() == 1:
        return 4*(ch[k] - 1)

def banana(sign, s, k): #The equation itself: choosing sign=+1 or -1 gives each
                         →of the two arcs
    polring=PolynomialRing(L, 'x,y')
    x, y=polring.0, polring.1
    A,B,C,D=s[0][0], s[0][1], s[1][0], s[1][1]
    return
    →sign*((B-C)*(1+x^2+y^2)+2*x*(A-D)-2*y*(B+C))-(1-x^2-y^2)*sqrt(H(k,s)-(A+D)^2+4*det(s))
```

```

##### Defining the equations
var('x,y')
sign1,sign2=-1,-1 #Choose the right couple of signs to point to the right
→component of each banana!
eq1=banana(sign1, s[2], 2) #Insert here the right number of sidepairing s[j]
→for this case!!!!!
eq2=banana(sign2, s[10], 1) #The third number is either 0, 1, or 2 and
→indicates which banana we want.
#Choose the right ones for this case!!!
show(eq1)
show(eq2)
#####

##### A bit of algebraic geometry here: J is the variety determined by both
→polynomials. We check first that
##### it has dimension 0: it consists of finitely many points. Finally we compute
→those points.
polring=PolynomialRing(L, 'x,y')
J = polring.ideal(eq1, eq2)
print J.dimension()
J.variety(L)

(5*u^5 - 19*u^4 - 9*u^3 - 2*u^2 - 6*u - 23)*x^2 + (5*u^5 - 19*u^4 - 9*u^3 - 2*u^2
→- 6*u - 23)*y^2 + ((4*u^5 - 16*u^4 - 4*u^3 - 16)*v)*x + (-7*u^5 + 26*u^4 +
→14*u^3 + 4*u^2 + 9*u + 30)*y + 2*u^5 - 15/2*u^4 - 4*u^3 - u^2 - 2*u - 15/2

(3*u^5 - 11*u^4 - 7*u^3 - 2*u^2 - 2*u - 13)*x^2 + (3*u^5 - 11*u^4 - 7*u^3 - 2*u^2
→- 2*u - 13)*y^2 + ((4*u^5 - 15*u^4 - 8*u^3 - 4*u - 17)*v)*x + (-5*u^5 + 19*u^4 +
→+ 9*u^3 + 4*u + 21)*y + 2*u^5 - 15/2*u^4 - 4*u^3 - u^2 - 2*u - 19/2

verbose 0 (3497: multi_polynomial_ideal.py, groebner_basis) Warning: falling
back to very slow toy implementation.
verbose 0 (1080: multi_polynomial_ideal.py, dimension) Warning: falling back to
very slow toy implementation.
0
verbose 0 (1080: multi_polynomial_ideal.py, dimension) Warning: falling back to
very slow toy implementation.
verbose 0 (2132: multi_polynomial_ideal.py, variety) Warning: falling back to
very slow toy implementation.
verbose 0 (3497: multi_polynomial_ideal.py, groebner_basis) Warning: falling
back to very slow toy implementation.
verbose 0 (3497: multi_polynomial_ideal.py, groebner_basis) Warning: falling
back to very slow toy implementation.

[28]: [{y: -1/8*u^4 + 1/2*u^3 + 1/2*u + 1/8, x: (5/28*u^5 - 11/14*u^4 + 3/7*u^2 +
11/28*u - 9/14)*v},
{y: -51/232*u^5 + 47/58*u^4 + 21/58*u^3 + 31/58*u^2 + 95/232*u + 55/58, x:

```

```
(39/406*u^5 - 395/812*u^4 + 13/58*u^3 + 72/203*u^2 + 101/203*u - 229/812)*v}]
```

9) We apply L_approx to the points we have obtained in order to determine which is the one that corresponds to the value we expected from our numerical experiments. If none of them seems to be correct we need to go back to 8) and change the signs.

```
[29]: #WARNING: The numerical experiments gave z~0.516-0.248i in this surface
for point in J.variety(L):
    print L_approx(point[x]),L_approx(point[y])
```

```
verbose 0 (1080: multi_polynomial_ideal.py, dimension) Warning: falling back to
very slow toy implementation.
verbose 0 (2132: multi_polynomial_ideal.py, variety) Warning: falling back to
very slow toy implementation.
verbose 0 (3497: multi_polynomial_ideal.py, groebner_basis) Warning: falling
back to very slow toy implementation.
verbose 0 (3497: multi_polynomial_ideal.py, groebner_basis) Warning: falling
back to very slow toy implementation.
0.515984860898360 -0.248485212697595
0.140546828416111 -1.08087246943746
```

10) We call zx and zy the real and imaginary part of the right intersection of bananas. The point $z = zx + izy$ is, still conjecturally, the center of a disc of a second (hidden) extremal packing.

```
[30]: zx, zy=J.variety(L)[0][x], J.variety(L)[0][y]
#Choose J.variety(L)[S] for the right value of S checking the output of 9)

show(zx) #Again having a look. Are they very large expressions in u?
show(zy)
```

```
verbose 0 (1080: multi_polynomial_ideal.py, dimension) Warning: falling back to
very slow toy implementation.
verbose 0 (2132: multi_polynomial_ideal.py, variety) Warning: falling back to
very slow toy implementation.
verbose 0 (3497: multi_polynomial_ideal.py, groebner_basis) Warning: falling
back to very slow toy implementation.
verbose 0 (3497: multi_polynomial_ideal.py, groebner_basis) Warning: falling
back to very slow toy implementation.
verbose 0 (1080: multi_polynomial_ideal.py, dimension) Warning: falling back to
very slow toy implementation.
verbose 0 (2132: multi_polynomial_ideal.py, variety) Warning: falling back to
very slow toy implementation.
verbose 0 (3497: multi_polynomial_ideal.py, groebner_basis) Warning: falling
back to very slow toy implementation.
```

verbose 0 (3497: multi_polynomial_ideal.py, groebner_basis) Warning: falling back to very slow toy implementation.

```
(5/28*u^5 - 11/14*u^4 + 3/7*u^2 + 11/28*u - 9/14)*v
-1/8*u^4 + 1/2*u^3 + 1/2*u + 1/8
```

11) Our numerical experiments suggest that the following center of the initial k-packing will have an important role:

$$w = \frac{u^2 - 1}{u^2 + 1} \left(\sin \frac{2\pi j}{N} - i \cos \frac{2\pi j}{N} \right)$$

for certain integer j (we need to get the right one in each case from the experiments).

(there is also an special case possible in which $w = 0$, the center of the unit disc).

We compute $\cos \frac{2\pi j}{N}$ as the value at $x = \cos \frac{\pi}{N}$ of the $2j$ -th Chebyshev polynomial. Then, we compute $\sin \frac{2\pi j}{N} = \sqrt{1 - \cos^2 \frac{2\pi j}{N}}$.

Important.- By the last expression we mean *the positive root*, but we don't know which of the two possible roots will Sage associate to the symbols $\sqrt{}$ and $-\sqrt{}$ applied to a given element of the field L. We need again to make use of L_approx to decide which one is the positive one (the warning is motivated for the fact that it may happen that the positive one is the one sage calls $-\sqrt{}$).

```
[31]: #w_is_center='yes' #leave uncommented the right line. w_is_center='yes' si w is
      ↪the center of the unit disc
w_is_center='no'

if w_is_center=='yes':
    wx=0
    wy=0
elif w_is_center=='no':
    j=2 #Enter here the right value for j in this case
    wy=-(u^2-1)*chebyshev_T(2*j,cs)/(u^2+1)
    wx=(u^2-1)*(sqrt(1-chebyshev_T(2*j,cs)^2))/(u^2+1)
    if L_approx(wx)<0: wx=-wx
    show(wx) #Looking at the expressions as polynomials in u or u and v
    show(wy)
    print L_approx(wx), L_approx(wy) #Numerical values
```

```
(1/4*u^4 - 1/2*u^3 - 2*u^2 - 3/2*u - 1/4)*v
-1/8*u^5 + 1/2*u^4 + 1/2*u^2 + 1/8*u + 1/2
0.700137725647924 -0.558341204756882
```

12) We define the function $g(z) = \frac{z-w}{\bar{w}z-1} = \frac{(z-w)(w\bar{z}-1)}{|w|^2|z|^2-2\operatorname{re}(w\bar{z})+1} = \frac{x+iy-wx-iwy}{(wx-iwy)(x+iy)-1}$, but rather as a function $\mathbb{R}^2 \rightarrow \mathbb{R}^2$. This is for technical reasons (L has no chosen embedding into the complex numbers, so we couldn't perform in principle products as $u \cdot i$ that we will later need).

We need to compute the point $p = \frac{g(zx+izy)}{1+\sqrt{1-|zx+izy|^2}}$, but the denominator is never in L because $1 - |zx + izy|^2$ is never a square. So we extend the field L to $K2 = L(w)$, where $w = \sqrt{1 - |zx + izy|^2}$.

Finally $m = g(p)$ may be, according to our numerical experiments, the fixed point of an order two elliptic element inducing an automorphism of the surface.

```
[32]: def g(x,y):
    denom=(x*wx+y*wy-1)^2+(y*wx-x*wy)^2
    num1=(x^2+y^2)*wx-x*(wx^2-wy^2)-2*y*wx*wy-x+wx
    num2=(x^2+y^2)*wy-2*x*wx*wy+y*(wx^2-wy^2)-y+wy
    return [num1/denom, num2/denom]

show((1-(g(zx,zy)[0])^2-(g(zx,zy)[1])^2)) #Inspecting how does w^2 look like

if (1-(g(zx,zy)[0])^2-(g(zx,zy)[1])^2).is_square():
    print 'Warning: apparently w IS in the field L' #Apparently this never
    →happens in our files
    #If it happens the last lines of this cell would produce an error, since
    →K2_approx
    #wouldn't be defined
else:
    Ring.<y> = PolynomialRing(L)
    K2.<omega> = L.extension(y^2-(1-(g(zx,zy)[0])^2-(g(zx,zy)[1])^2))
    px=g(zx,zy)[0]/(1+omega)
    py=g(zx,zy)[1]/(1+omega)
    mx=g(px,py)[0]
    my=g(px,py)[1]
    show(mx) #These are the coordinates of the fixed point of tau as elements of
    →the field L.
    show(my)
    #Again choosing a precise embedding of K2 into the complex numbers
    #This is not really needed, but allows the computation of numerical values
    →for mx, my
    j=0
    test=0
    while test!=1:
        K2_approx = K2.embeddings(CC)[j]
        if abs(CC(sqrt(u^4+u^2+1)/(u^2+1))-K2_approx(v))<10^-3 and
        →abs(CC(u)-K2_approx(u))<10^(-3) and K2_approx(omega)>0: test+=1
```

```

j+=1
print K2_approx(u),K2_approx(v), K2_approx(omega) #This line must give the
→values we knew in advance (numerical verification)

```

```
print K2_approx(mx), K2_approx(my)
```

```

u^5 - 4*u^4 - u^3 - 2*u + 1
((-17/254*u^5 + 29/127*u^4 + 41/127*u^3 - 50/127*u^2 + 15/254*u + 39/127)*v)*omega
→+ (20/127*u^5 - 129/254*u^4 - 89/127*u^3 + 28/127*u^2 - 55/127*u - 191/254)*v
(19/254*u^5 - 167/508*u^4 - 1/127*u^3 + 26/127*u^2 + 43/254*u - 137/508)*omega - u
→-67/508*u^5 + 251/508*u^4 + 57/254*u^3 + 21/127*u^2 + 89/508*u + 285/508
4.25917178742060 0.974927912486875 0.711365432959471
0.619750016742053 -0.459297917965105

```

13) We compute the coordinates of $\bar{\tau}$

[33] :

```

tau00=(-2*mx)/(1-mx^2-my^2)
tau01=(2*my+1+mx^2+my^2)/(1-mx^2-my^2)
tau10=(2*my-1-mx^2-my^2)/(1-mx^2-my^2)
tau11=(2*mx)/(1-mx^2-my^2)
tau=matrix(2,[tau00,tau01,tau10,tau11])
show(tau) # Once more we are curious. The entries lie in K2, but: how do they
→look like?

```

```

[((-25/26*u^5 + 47/13*u^4 + 22/13*u^3 + 10/13*u^2 + 29/26*u + 53/13)*v)*omega
→(-97/52*u^5 + 181/26*u^4 + 97/26*u^3 + 29/26*u^2 + 115/52*u + 105/13)*omega]
[ (27/52*u^5 - 2*u^4 - 25/26*u^3 - 1/26*u^2 - 1/4*u - 59/26)*omega
→((25/26*u^5 - 47/13*u^4 - 22/13*u^3 - 10/13*u^2 - 29/26*u - 53/13)*v)*omega]

```

14) We create a list conj containing what our numerical experiments suggested that should be c_j , the conjugate by τ of the side-pairing s_j , for each j . We finally check that $\tau s_j \tau^{-1} c_j^{-1}$ is trivial.

[34] :

```

conj=['Zero'] #Create this list with the data from the numerical experiments!!!!
conj.append(s[2]^(-1)*s[6]*s[7]^(-1)*s[2]) #1
conj.append(s[2]^(-1)*s[7]) #2
conj.append(s[2]^(-1)*s[6]*s[7]^(-1)*s[2]) #3
conj.append(s[2]^(-1)*s[7]) #4
conj.append(s[2]^(-1)*s[1]^(-1)*s[6]^(-1)*s[2]) #5
conj.append(s[2]^(-1)*s[7]*s[3]*s[2]) #6
conj.append(s[2]^(-1)*s[7]*s[2]) #7
conj.append(s[2]^(-1)*s[3]*s[2]) #8
conj.append(s[2]^(-1)*s[14]^(-1)*s[2]) #9

```

```

conj.append(s[2]^(-1)*s[14]^(-1)*s[11]*s[2]) #10
conj.append(s[2]^(-1)*s[14]^(-1)*s[11]^(-1)*s[14]*s[2]) #11
conj.append(s[2]^(-1)*s[16]^(-1)*s[12]^(-1)*s[16]*s[2]) #12
conj.append(s[2]^(-1)*s[14]^(-1)*s[11]*s[2]*s[7]^(-1)*s[2]) #13
conj.append(s[2]^(-1)*s[14]^(-1)*s[7]^(-1)*s[2]) #14
conj.append(s[2]^(-1)*s[16]^(-1)*s[14]*s[2]) #15
conj.append(s[2]^(-1)*s[16]^(-1)*s[7]^(-1)*s[2]) #16
conj.append(s[2]^(-1)*s[16]^(-1)*s[4]^(-1)*s[11]^(-1)*s[14]*s[2]) #17
conj.append(s[2]^(-1)*s[16]^(-1)*s[4]^(-1)*s[11]^(-1)*s[14]*s[2]) #18

for j in [1..len(s)-1]:
    show(tau*s[j]*tau^(-1)*conj[j]^(-1))

```

```

[1 0]
[0 1]

[-1 0]
[ 0 -1]

[-1 0]
[ 0 -1]

[1 0]
[0 1]

[1 0]
[0 1]

[-1 0]
[ 0 -1]

[1 0]
[0 1]

[1 0]
[0 1]

[-1 0]
[ 0 -1]

[-1 0]
[ 0 -1]

[1 0]
[0 1]

[1 0]
[0 1]

[1 0]
[0 1]

```

$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$

$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$

$\begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}$

$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

$\begin{bmatrix} \quad & \quad \end{bmatrix} :$