Content and output of `QuotientCurves.sage`

Nicolas Müller, Richard Pink

November 17, 2017

This is a formatted version of the Sage code in the file `QuotientCurves.sage`, which can be found at [1], together with its output. This worksheet belongs to [2].

```
# Date: November 17, 2017

# This function replaces x^power by y in the given polynomial f
def replacePowerOfVariablePoly(f, x, power, y):
    coeffs = (f*x^0).coefficients(sparse = False)
    assert all([c == 0 or mod(l,power) == 0 for c,l in
            zip(coeffs, range(len(coeffs)))])
    return sum([x^(l/power)*c for c,l in zip(coeffs, range(len(coeffs)))
            if mod(l,power)==0])

# This function replaces x^power by y in the given rational function or
# polynomial f
def replacePowerOfVariable(f, x, power, y):
    num = f.numerator()
    denom = f.denominator()
    return replacePowerOfVariablePoly(num, x, power, y) / \
           replacePowerOfVariablePoly(denom, x, power, y)

# This function applies the Moebius transform corresponding to the
# matrix T to x.
def moeb(T,x):
    return (T[0][0]*x+T[0][1])/(T[1][0]*x+T[1][1])

# This function returns a matrix that corresponds to the given Moebius
# transform T in the variable x.
def matrixFromMoeb(T,x):
    numCoeffs = (T.numerator()*x^0).coefficients(sparse = False)
    denomCoeffs = (T.denominator()*x^0).coefficients(sparse = False)
    assert len(numCoeffs) <= 2 and len(denomCoeffs) <= 2
    if len(numCoeffs) == 2:
        a = numCoeffs[1]
    else: a = 0
    b = numCoeffs[0]
    if len(denomCoeffs) == 2:
        c = denomCoeffs[1]
    else: c = 0
    d = denomCoeffs[0]
    return Matrix([[a,b],[c,d]])
```

```
# This function calculates the quotient of the hyperelliptic curve X:
# y^2 = f(x) by the subgroups generated by automorphisms T' that map
# to T in the reduced automorphism group.
#
# INPUT:
# - T: A matrix that corresponds to a Moebius transform in the reduced
#      automorphism group of X, in the coordinates given by f.
# - f: A separable polynomial such that y^2 = f(x) is an affine equation
#      of X.
# - K: A field extension of the rationals over which T and f are defined
#      and over which T can be diagonalized.
# OUTPUT:
# If calculating the quotient succeeds, the output is a triple
# (f_list, S, n), where
#   - f_list is a list of one or two polynomials such that the quotients
#     of X by lifts of T to Aut(X) are given by affine equations
#     y^2 = fnew(x) for fnew in f_list.
#   - S is a matrix such that x -> moeb(S, x) is the coordinate
#     transform applied to P^1 before taking the quotient.
#   - n is the order of the element of the reduced automorphism group
#     of X that corresponds to the matrix T.
# If calculating the quotient does not succeed because the quotient has
# genus 0, None is returned.
def quotientByCyclicGroup(T, f, K):
    R.<x> = K[]
    f = R(f)

    # We check that T lifts to an automorphism of X.
    moebT = moeb(T,x)
    tempSubs = f.substitute(x = moebT) * \
               moebT.denominator()^((ceil(f.degree()/2))*2)
    assert tempSubs == tempSubs.numerator()
    tempSubs = tempSubs.numerator()
    assert tempSubs/tempSubs.leading_coefficient() == \
           f/f.leading_coefficient()

    # We calculate a coordinate transform of P^1 such that T acts as a
    # rotation fixing 0 and infinity.
    D, S = T.eigenmatrix_left()
    Si = S^(-1)
    moebSi = moeb(Si,x)

    # n = the order of the Moebius transform corresponding to T
    n = (D[0][0]/D[1][1]).multiplicative_order()

    # After the coordinate transform, the equation of the curve is
    # y^2 = h(x).
    h = f.substitute(x = moebSi) * \
        moebSi.denominator()^((ceil(f.degree()/2))*2)
    assert h == h.numerator() # We check that h is indeed a polynomial ..
    h = h.numerator()
    h /= h.leading_coefficient() # .. and make h monic.
```

```
    # We define k as in the proof of Proposition 5.1.
    if h(0) == 0: k = 1
    else: k = 0

    # We define g as in the proof of Proposition 5.1.
    g = replacePowerOfVariable(h/x^k, x, n , x)
    assert g == g.numerator()
    g = g.numerator()

    if mod(n, 2) == 0:
        if k == 1:
            print "Quotient␣has␣genus␣0!"
            return None
        else: # k == 0
            # An affine equation of the quotient is y^2 = fnew1(x) or
            # y^2 = fnew2(x), depending on the lift of T to Aut(X).
            fnew1 = g
            fnew2 = x * g
            return ([fnew1, fnew2], S, n)
    else: # n odd
        # An affine equation of the quotient is y^2 = fnew(x).
        fnew = x^k * g
        return ([fnew], S, n)


# After setting up the functions, we calculate the quotients of the
# curves as stated in Proposition 5.1.

# K is the number field generated by a primitive 120-th root of unity e120.
K.<e120> = CyclotomicField(120)
# We define some more roots of unity that we need later.
e4=e120^30
e5=e120^24
e12=e120^10
e15=e120^8
e20=e120^6

R.<x> = K[]
R2.<x2,y2> = K[]

# Define the polynomials as in the paper.
s4 = x^8+14*x^4+1
t4 = x*(x^4-1)
r4 = x^12-33*x^8-33*x^4+1
s5 = x*(x^10+11*x^5-1)
r5 = x^20-228*x^15+494*x^10+228*x^5+1
t5 = x^30+522*x^25-10005*x^20-10005*x^10-522*x^5+1

################################################## X6
f = s4
T = Matrix([[-1,0],[0,1]])

# We take the quotient by subgroups of Aut(X6) that map to the group
# generated by moeb(T,x) in the reduced automorphism group.
```

```
fT, S1, n1 = quotientByCyclicGroup(T, f, K)
# There are two quotients, corresponding to different lifts of moeb(T, x)
# to Aut(X6). We pick the first one.
fT = fT[0]
print "X6 quotient equation: "+str(fT)+" of genus "+ \
       str(HyperellipticCurve(fT).genus()) # x^4 + 14*x^2 + 1
print "j-invariant: "+\
      str(QQ(Jacobian(fT.substitute(x = x2)-y2^2).j_invariant()).factor())
# Output: 2^4 * 3^-2 * 13^3
```

Output:

```
X6 quotient equation: x^4 + 14*x^2 + 1 of genus 1
j-invariant: 2^4 * 3^-2 * 13^3
```

```
################################################# X8
f = s4*t4
# The following is a composition series of S3 (as a subgroup of the
# reduced automorphism group of X8) such that all subquotients are cyclic:
# 1 < <T1> < <T1, T2> = S3

# The matrices T1, T2 correspond to the elements of reduced automorphism
# group of X8 used in the above composition series.
T1 = Matrix([[e4, 1],[e4, -1]])
T2 = Matrix([[-1, 1],[1,1]])

# We first take the quotient by a subgroup of Aut(X8) that maps to the
# subgroup generated by the Moebius transform induced by T1 in the
# reduced automorphism group.
fT1, S1, n1 = quotientByCyclicGroup(T1, f, K)
# Since the order of moeb(T1,x) is odd, there is only one possible
# quotient curve.
fT1 = fT1[0]

# We apply the change of coordinates to T2 that was used when taking the
# quotient of X8 by a lift of T1.
T2 = S1*T2*S1^(-1)
T2moeb = moeb(T2,x)

# T2 normalizes T1. Therefore T2 induces an automorphism on the quotient
# y^2 = fT1(x).
# After the change of coordinates the quotient map X8 -> X8/<lift of T1>
# is given, on P^1, by x -> x^n1.
# T2 acts on P^1/<T1> by T2quotmoeb = T2moeb(x^(1/n1))^n1.
T2quotmoeb = replacePowerOfVariable((T2moeb)^n1, x, n1, x)
T2quotmat = matrixFromMoeb(T2quotmoeb, x)

# We take the quotient of X8/<lift of T1> by a lift of T2quot to
# Aut(X8/<lift of T1>)
fT1T2, S2, n2 = quotientByCyclicGroup(T2quotmat, fT1, K)

# There are two possible quotients (elliptic curves with complex
# conjugate j-invariants) corresponding to different lifts of <T1,T2>
# to Aut(X8). We pick the first one.
fT1T2 = fT1T2[0]
```

```
print "X8␣quotient␣equation:␣"+str(fT1T2)+"␣of␣genus␣"+ \
      str(HyperellipticCurve(fT1T2).genus())
# Output: x^3
#          + (26/81*e120^25 + 26/81*e120^15 - 26/81*e120^5 - 167/81)*x^2
#          + (-1820/2187*e120^25 - 1820/2187*e120^15 + 1820/2187*e120^5
#              + 3833/2187)*x
#          + 1118/2187*e120^25 + 1118/2187*e120^15
#          - 1118/2187*e120^5 - 1511/2187
sqrtm2 = e120^25 + e120^15 - e120^5
# We put the curve in Legendre form:
fT1T2_2  = fT1T2.substitute(x = (1-x)+x*(-56/81*sqrtm2 + 17/81))/ \
           (-336896/531441*sqrtm2 + 942080/531441)
K2.<sqrtm2> = K.subfield(sqrtm2,"sqrtm2")[0]
print "X8␣quotient,␣simplified␣equation:␣"+ \
       str((fT1T2_2.change_ring(K2)).factor())
# Output: (x - 1) * x * (x + 1/4*sqrtm2 + 1/4)
print "j-invariant:␣"+str(Jacobian((fT1T2_2.substitute(x = x2)-y2^2). \
                                          change_ring(K2)).j_invariant())
# Output: -855712/729*sqrtm2 + 467888/729
```

Output:

```
X8 quotient equation: x^3 + (26/81*e120^25 + 26/81*e120^15 - 26/81*e120^5 -↙
    167/81)*x^2 + (-1820/2187*e120^25 - 1820/2187*e120^15 + 1820/2187*e120↙
   ^5 + 3833/2187)*x + 1118/2187*e120^25 + 1118/2187*e120^15 - 1118/2187*↙
   e120^5 - 1511/2187 of genus 1
X8 quotient, simplified equation: (x - 1) * x * (x + 1/4*sqrtm2 + 1/4)
j-invariant: -855712/729*sqrtm2 + 467888/729
```

```
################################################## X10
f = r4*s4
T = Matrix([[e4,0],[0,1]])

# We take the quotient by subgroups of Aut(X10) that map to the group
# generated by moeb(T,x) in the reduced automorphism group.
fT, S, n = quotientByCyclicGroup(T, f, K)
# There are two quotients, corresponding to different lifts of moeb(T,x)
# to Aut(X10). We pick the first one.
fT = fT[0]
print "X10␣quotient␣equation:␣"+str(fT)+"␣of␣genus␣"+ \
      str(HyperellipticCurve(fT).genus())
# Output: x^5 - 19*x^4 - 494*x^3 - 494*x^2 - 19*x + 1
```

Output:

```
X10 quotient equation: x^5 - 19*x^4 - 494*x^3 - 494*x^2 - 19*x + 1 of genus↙
    2
```

```
################################################## X11
f = r4*s4*t4
T = Matrix([[e4, 1],[e4, -1]])

# We take the quotient by a subgroup of Aut(X11) that maps to the group
# generated by moeb(T,x) in the reduced automorphism group.
```

```
fT, S, n = quotientByCyclicGroup(T, f, K)
# Since the order of moeb(T,X) is odd, there is only one possible
# quotient curve.
fT = fT[0]
print "X11␣quotient␣equation:␣"+str(fT)+"␣of␣genus␣"+ \
      str(HyperellipticCurve(fT).genus())
# Output: x^9
#         + (-225/4*e120^30 + 225/2*e120^10 - 375/4)*x^8
#         + (-4875/2*e120^30 + 4875*e120^10 - 4225)*x^7
#         + (247095/4*e120^30 - 247095/2*e120^10 + 427975/4)*x^6
#         + (12844095/4*e120^30 - 12844095/2*e120^10 + 22246625/4)*x^4
#         + (13177125/2*e120^30 - 13177125*e120^10 + 11411725)*x^3
#         + (-31005225/4*e120^30 + 31005225/2*e120^10 - 53702625/4)*x^2
#         + (-2107560*e120^30 + 4215120*e120^10 - 3650401)*x
sqrt3 = 2*e12 - e12^3
# We simplify the equation with a substitution:
fT_2 = fT.substitute(x = x*(2-sqrt3)^(2))/((2-sqrt3)^(9*2)*(1+sqrt3))*4
# We move the root (1+sqrt3)/4 to 1 and make the polynomial monic again.
fT_3 = fT_2.substitute(x = x*(1+sqrt3)/4)*4096/(97+56*sqrt3)
# fT is now rational. We move some of the rational roots to 0,1,infinity
# and make the polynomial monic.
fT_4 = fT_3.substitute(x = 8*(1-x)/(x+8))*(x+8)^10/(-52242776064)
print "X11␣quotient␣equation␣after␣a␣coordinate␣change:␣"+ \
      str(QQ[x](fT_4).factor())
```

Output:

```
X11 quotient equation: x^9 + (-225/4*e120^30 + 225/2*e120^10 - 375/4)*x^8 +↙
    (-4875/2*e120^30 + 4875*e120^10 - 4225)*x^7 + (247095/4*e120^30 - ↙
   247095/2*e120^10 + 427975/4)*x^6 + (12844095/4*e120^30 - 12844095/2*↙
   e120^10 + 22246625/4)*x^4 + (13177125/2*e120^30 - 13177125*e120^10 + ↙
   11411725)*x^3 + (-31005225/4*e120^30 + 31005225/2*e120^10 - 53702625/4)↙
   *x^2 + (-2107560*e120^30 + 4215120*e120^10 - 3650401)*x of genus 4
X11 quotient equation after a coordinate change: (x - 1) * x * (x + 8) * (x↙
   ^2 + 8) * (x^2 + 4*x - 8) * (x^2 + 8*x - 8)
```

```
################################################## X12
f = s5
T = Matrix([[e5,0],[0,1]])

# We take the quotient by a subgroup of Aut(X12) that maps to the group
# generated by moeb(T,x) in the reduced automorphism group.
fT, S, n = quotientByCyclicGroup(T, f, K)
# Since the order of moeb(T,x) is odd, there is only one possible
# quotient curve.
fT = fT[0]
print "X12␣quotient␣equation:␣"+str(fT)+"␣of␣genus␣"+ \
      str(HyperellipticCurve(fT).genus()) # x^3 - 11*x^2 - x
print "X12␣quotient␣equation,␣alternate␣form:␣"+ \
      str(-fT.substitute(x=-x)) # x^3 + 11*x^2 - x
print "j-invariant:␣"+ \
      str(QQ(Jacobian(fT.substitute(x = x2)-y2^2).j_invariant()).factor())
# Output: 2^14 * 5^-3 * 31^3
```

Output:

```
X12 quotient equation: x^3 - 11*x^2 - x of genus 1
X12 quotient equation, alternate form: x^3 + 11*x^2 - x
j-invariant: 2^14 * 5^-3 * 31^3
```

```
################################################### X13
f = r5
T = Matrix([[e5,0],[0,1]])

# We take the quotient by a subgroup of Aut(X13) that maps to the group
# generated by moeb(T,x) in the reduced automorphism group.
fT, S, n = quotientByCyclicGroup(T, f, K)
# Since the order of moeb(T,x) is odd, there is only one possible
# quotient curve.
fT = fT[0]
print "X13 quotient equation: "+str(fT)+" of genus "+ \
       str(HyperellipticCurve(fT).genus())
# Output: x^4 + 228*x^3 + 494*x^2 - 228*x + 1
print "X13 quotient equation, alternate form: "+str(fT.substitute(x=-x))
# Output: x^4 - 228*x^3 + 494*x^2 + 228*x + 1
print "j-invariant: "+ \
      str(QQ(Jacobian(fT.substitute(x = x2)-y2^2).j_invariant()).factor())
# Output: 2^17 * 3^-2
```

Output:

```
X13 quotient equation: x^4 + 228*x^3 + 494*x^2 - 228*x + 1 of genus 1
X13 quotient equation, alternate form: x^4 - 228*x^3 + 494*x^2 + 228*x + 1
j-invariant: 2^17 * 3^-2
```

```
################################################### X15
f = r5*s5

# A composition series of a subgroup H of the reduced automorphism group
# of X15 isomorphic to A4 such that all subquotients are cyclic is given
# by 1 < <T1> < <T1,T2> < <T1,T2,T3> = H .

T1 = Matrix([[ 0, -e5 ], [ e5^4, 0 ]])
T2 = Matrix([[ 2*e5-e5^2+e5^3-2*e5^4, e5+2*e5^2+3*e5^3-e5^4 ],
             [ e5-3*e5^2-2*e5^3-e5^4, -2*e5+e5^2-e5^3+2*e5^4 ]])
T3 = Matrix([[1, -e5^2-e5^3], [e5+e5^2+e5^3, -e5^2]])

# We first take the quotient by groups H1 in Aut(X15) that map to <T1>
# in the reduced automorphism group.
fT1, S1, n1 = quotientByCyclicGroup(T1, f, K)
# There are two quotients, corresponding to different lifts of T1
# to Aut(X15). We pick the first one.
fT1 = fT1[0]

# We apply the change of coordinates to T2 that was used when taking the
# quotient of X15 by a lift of T1.
T2 = S1*T2*S1^(-1)
T2moeb = moeb(T2,x)
# The Moebius transform T2moeb induces a Moebius transform T2quot on
# P^1/<T1> \cong P^1, analogously to the case of X8.
T2quotmoeb = replacePowerOfVariable((T2moeb)^n1, x, n1, x)
```

```
T2quotmat = matrixFromMoeb(T2quotmoeb, x)

# Next we take the quotient by subgroups of Aut(X15/H1) inducing T2quot
# on the reduced automorphism group of X15/H1.
fT1T2, S2, n2 = quotientByCyclicGroup(T2quotmat, fT1, K)
# There are two quotients, corresponding to different lifts of
# moeb(T2quot,x) to Aut(X15/H1). We pick the first one.
fT1T2 = fT1T2[0]

# We apply the first change of coordinates to T3.
T3 = S1*T3*S1^(-1)

T3moeb = moeb(T3,x)
S2moeb = moeb(S2,x)
Si2moeb = moeb(S2^(-1),x)

# After the first change of coordinates, the map P^1 -> P^1/<T1,T2> is
# given by phi: x -> S2moeb(x^n1)^n2. Therefore T3 acts on P^1/<T1,T2> by
# phi(T3moeb(phi^(-1)(x))). This is well-defined since <T1,T2> is
# normal in <T1,T2,T3>.
T3tempmoeb = replacePowerOfVariable(S2moeb(T3moeb^(n1))^(n2), x, n2, x)
T3quotmoeb = replacePowerOfVariable(T3tempmoeb(Si2moeb), x, n1 , x)
T3quotmat = matrixFromMoeb(T3quotmoeb, x)

# Finally, we take the quotient by an automorphism of y^2 = fT1T2(x) that
# induces T3quot in the reduced automorphism group of the curve
# y^2 = fT1T2(x) and obtain an elliptic curve.
fT1T2T3, S3, n3 = quotientByCyclicGroup(T3quotmat, fT1T2, K)
# Since the order of moeb(T3,x) is odd, there is only one possible
# quotient curve.
fT1T2T3 = fT1T2T3[0]
print "X15␣quotient␣equation:␣"+str(fT1T2T3)+"␣of␣genus␣"+ \
      str(HyperellipticCurve(fT1T2T3).genus())
# Output: x^3
#         + (93/512*e120^28 - 93/512*e120^20 - 93/256*e120^16
#          - 93/512*e120^12 - 93/512*e120^8 + 93/256*e120^4 - 223/512)*x^2
#         + (-13485/32768*e120^28 + 13485/32768*e120^20
#           + 13485/16384*e120^16 + 13485/32768*e120^12
#           + 13485/32768*e120^8 - 13485/16384*e120^4 - 433/32768)*x
print "j-invariant:␣"+ \
str(QQ(Jacobian(fT1T2T3.substitute(x = x2)-y2^2).j_invariant()).factor())
# Output: 2^2 * 3^-3 * 19^3
fT1T2T3_2 = (-EllipticCurve_from_j(2^2 * 3^-3 * 19^3). \
              defining_polynomial())(x,0,1)
fT1T2T3_3 = fT1T2T3_2.substitute(x = x+2)
print "X15␣quotient␣equation,␣alternate␣form:␣"+str(fT1T2T3_3)
# Output: x^3 + 5*x^2 + 40*x
```

Output:

```
X15 quotient equation: x^3 + (93/512*e120^28 - 93/512*e120^20 - 93/256*e120↙
   ^16 - 93/512*e120^12 - 93/512*e120^8 + 93/256*e120^4 - 223/512)*x^2 + ↙
   (-13485/32768*e120^28 + 13485/32768*e120^20 + 13485/16384*e120^16 + ↙
   13485/32768*e120^12 + 13485/32768*e120^8 - 13485/16384*e120^4 - ↙
   433/32768)*x of genus 1
```

```
j-invariant: 2^2 * 3^-3 * 19^3
X15 quotient equation, alternate form: x^3 + 5*x^2 + 40*x
```

```
################################################ X16
f = s5*t5
T = Matrix([[e5,0],[0,1]])

# We take the quotient by a subgroup of Aut(X16) that maps to the group
# generated by moeb(T,x) in the reduced automorphism group.
fT, S, n = quotientByCyclicGroup(T, f, K)
# Since the order of moeb(T,x) is odd, there is only one possible
# quotient curve.
fT = fT[0]
print "X16 quotient equation: "+str(fT)+" of genus "+ \
       str(HyperellipticCurve(fT).genus())
# Output: x^9 - 533*x^8 - 4264*x^7 + 110577*x^6 + 110577*x^4 + 4264*x^3
#         - 533*x^2 - x
print "X16 quotient equation, alternate form: "+ \
       str((-fT.substitute(x = -x)).change_ring(QQ).factor())
# Output: x * (x^2 + 1) * (x^2 + 11*x - 1)
#         * (x^4 + 522*x^3 - 10006*x^2 - 522*x + 1)
```

Output:

```
X16 quotient equation: x^9 - 533*x^8 - 4264*x^7 + 110577*x^6 + 110577*x^4 +
    4264*x^3 - 533*x^2 - x of genus 4
X16 quotient equation, alternate form: x * (x^2 + 1) * (x^2 + 11*x - 1) * (
   x^4 + 522*x^3 - 10006*x^2 - 522*x + 1)
```

```
################################################ X17
f = r5*t5
T = Matrix([[e5,0],[0,1]])

# We take the quotient by a subgroup of Aut(X17) that maps to the group
# generated by moeb(T,x) in the reduced automorphism group.
fT, S, n = quotientByCyclicGroup(T, f, K)
# Since the order of moeb(T,x) is odd, there is only one possible
# quotient curve.
fT = fT[0]
print "X17 quotient equation: "+str(fT)+" of genus "+ \
       str(HyperellipticCurve(fT).genus())
# Output: x^10 - 294*x^9 - 128527*x^8 - 2539236*x^7 - 4833458*x^6
#         - 4833458*x^4 + 2539236*x^3 - 128527*x^2 + 294*x + 1
print "X17 quotient equation, alternate form: "+ \
      str((fT.substitute(x = -x)).change_ring(QQ).factor())
# Output: (x^2 + 1) * (x^4 - 228*x^3 + 494*x^2 + 228*x + 1)
#         * (x^4 + 522*x^3 - 10006*x^2 - 522*x + 1)
```

Output:

```
X17 quotient equation: x^10 - 294*x^9 - 128527*x^8 - 2539236*x^7 - 4833458*
   x^6 - 4833458*x^4 + 2539236*x^3 - 128527*x^2 + 294*x + 1 of genus 4
X17 quotient equation, alternate form: (x^2 + 1) * (x^4 - 228*x^3 + 494*x^2
    + 228*x + 1) * (x^4 + 522*x^3 - 10006*x^2 - 522*x + 1)
```

```
################################################## X18
f = r5*s5*t5
T = Matrix([[e5,0],[0,1]])

# We take the quotient by a subgroup of Aut(X18) that maps to the group
# generated by moeb(T,x) in the reduced automorphism group.
fT, S, n = quotientByCyclicGroup(T, f, K)
# Since the order of moeb(T,x) is odd, there is only one possible
# quotient curve.
fT = fT[0]
print "X18 quotient equation: "+str(fT)+" of genus "+ \
      str(HyperellipticCurve(fT).genus())
# Output: x^13 - 305*x^12 - 125294*x^11 - 1125145*x^10 + 23226665*x^9
#          + 55707274*x^8 + 55707274*x^6 - 23226665*x^5 - 1125145*x^4
#          + 125294*x^3 - 305*x^2 - x
print "X18 quotient equation, alternate form: "+ \
      str((-fT.substitute(x = -x)).change_ring(QQ).factor())
# Output: x * (x^2 + 1) * (x^2 + 11*x - 1)
#          * (x^4 - 228*x^3 + 494*x^2 + 228*x + 1)
#          * (x^4 + 522*x^3 - 10006*x^2 - 522*x + 1)
```

Output:

```
X18 quotient equation: x^13 - 305*x^12 - 125294*x^11 - 1125145*x^10 + 
    23226665*x^9 + 55707274*x^8 + 55707274*x^6 - 23226665*x^5 - 1125145*x^4
     + 125294*x^3 - 305*x^2 - x of genus 6
X18 quotient equation, alternate form: x * (x^2 + 1) * (x^2 + 11*x - 1) * (
    x^4 - 228*x^3 + 494*x^2 + 228*x + 1) * (x^4 + 522*x^3 - 10006*x^2 - 
    522*x + 1)
```

References

[1] Worksheets for this paper: URL: https://people.math.ethz.ch/~pink/ftp/MuellerPink2017/

[2] Müller, N., Pink, R.: *Hyperelliptic Curves with Many Automorphisms*. Preprint 2017.