```
# Case \theta_0=3

var('A0,A1,A2,A3,A4,A5,A6,B0,B1,B2,B3,B5,B6,B7,B8,B9,B10,C0,C1,C3,C4
,C5,D0,D1,D2,D3,D4,D5,E0,E1,E2,E3,E4,E5,E6,lambda_1,lambda_2,lambda_
3,lambda_4,alpha0,alpha1,alpha2,alpha3,alpha4,alpha5,alpha6,alpha7,a
lpha8,alpha9,alpha10,alpha11,alpha12,alpha13,alpha14,alpha15,alpha16
,alpha17,alpha18,alpha19,alpha20,alpha21,alpha22,alpha23,E_00,E_10,E
_01,E_02,E_03,E_04,E_05,E_15,E_06,E_07,E_08,E_18,E_09,gamma2,gamma3,
gamma4,zeta0,zeta1,zeta2,zeta3,zeta4',domain='complex'),var('theta_0
,beta,gamma0,gamma1,gamma5,delta,C2,B4',domain='positive')

# the constant C2 becomes real in this particular case

n=theta_0
a0=A0
a0b=conjugate(A0)
b0=B0
b0b=conjugate(B0)
c0=C0
c0b=conjugate(C0)
d0=D0
d0b=conjugate(D0)
a1=A1
a1b=conjugate(A1)
b1=B1
b1b=conjugate(B1)
c1=C1
c1b=conjugate(C1)
d1=D1
d1b=conjugate(D1)
a2=A2
a2b=conjugate(A2)
b2=B2
b2b=conjugate(B2)
c2=C2
c2b=conjugate(C2)
d2=D2
d2b=conjugate(D2)
a3=A3
a3b=conjugate(A3)
b3=B3
b3b=conjugate(B3)
c3=C3
c3b=conjugate(C3)
d3=D3
d3b=conjugate(D3)
a4=A4
a4b=conjugate(A4)
b4=B4
b4b=conjugate(B4)
c4=C4
c4b=conjugate(C4)
d4=D4
d4b=conjugate(D4)
```

```
a5=A5
a5b=conjugate(A5)
b5=B5
b5b=conjugate(B5)
c5=C5
c5b=conjugate(C5)
d5=D5
d5b=conjugate(D5)
a6=A6
a6b=conjugate(A6)
b6=B6
b6b=conjugate(B6)
b7=B7
b7b=conjugate(B7)
b8=B8
b8b=conjugate(B8)
b9=B9
b9b=conjugate(B9)
b10=B10
b10b=conjugate(B10)
alpha0b=conjugate(alpha0)
alpha1b=conjugate(alpha1)
alpha2b=conjugate(alpha2)
alpha3b=conjugate(alpha3)
alpha4b=conjugate(alpha4)
alpha5b=conjugate(alpha5)
alpha6b=conjugate(alpha6)
alpha7b=conjugate(alpha7)
alpha8b=conjugate(alpha8)
alpha9b=conjugate(alpha9)
alpha10b=conjugate(alpha10)
alpha11b=conjugate(alpha11)
alpha12b=conjugate(alpha12)
alpha13b=conjugate(alpha13)
alpha14b=conjugate(alpha14)
alpha15b=conjugate(alpha15)
alpha16b=conjugate(alpha16)
alpha17b=conjugate(alpha17)
alpha18b=conjugate(alpha18)
alpha19b=conjugate(alpha19)
alpha20b=conjugate(alpha20)
alpha21b=conjugate(alpha21)
alpha22b=conjugate(alpha22)
alpha23b=conjugate(alpha23)
zeta0b=conjugate(zeta0)
zeta1b=conjugate(zeta1)
zeta2b=conjugate(zeta2)
e00=E_00
e10=E_10
e01=E_01
e02=E_02
e03=E_03
e04=E_04
e05=E_05
```

```
e15=E_15
e06=E_06
e07=E_07
e08=E_08
e18=E_18
e09=E_09
e0=E0
e0b=conjugate(E0)
e1=E1
e1b=conjugate(E1)
e2=E2
e2b=conjugate(E2)
e3=E3
e3b=conjugate(E3)
e4=E4
e4b=conjugate(E4)
e5=E5
e5b=conjugate(E5)
e6=E6
e6b=conjugate(E6)

def delete_zeroes(v):
    m=matrix(SR,v.nrows(),v.ncols())
    n=0
    for i in range(v.nrows()): # We first remove the zeroes
coefficients
        if v[i,0].is_zero():
            n=n+1
        else:
            for j in range(v.ncols()):
                m[i-n,j]=v[i,j]
    return m.submatrix(0,0,v.nrows()-n,v.ncols())

def factor_simplify(v):
    m=matrix(SR,v.nrows(),v.ncols())
    n=0
    for i in range(v.nrows()):
        if bool(v[i,0].is_zero()):
            n=n+1
        else:
            for j in range(v.ncols()):
                m[i-n,j]=v[i,j]
    m=m.submatrix(0,0,v.nrows()-n,v.ncols())
    if bool(m.ncols()==4):
        n=0
        mlength=m.nrows()
        for i in range(mlength):
            if bool(m[i,0].is_zero()):
                n=n
            else:
                for j in range(mlength-i-1):
                    if bool((m[i,1]-m[i+1+j,1]).is_zero()) &
bool((m[i,2]-m[i+1+j,2]).is_zero()) & bool((m[i,3]-
m[i+1+j,3]).is_zero()):
```

```
                        m[i,0]=m[i,0]+m[i+1+j,0]
                        m[i+1+j,0]=0
                        n=n+1
                    else:
                        n=n
        if bool(n==0):
            return m
        else:
            return delete_zeroes(m)
    else:
        n=0
        mlentgh=m.nrows()
        for i in range(mlentgh):
            if bool(m[i,0].is_zero()):
                n=n
            else:
                for j in range(mlentgh-i-1):
                    if bool((m[i,1]-m[i+1+j,1]).is_zero()) &
bool((m[i,2]-m[i+1+j,2]).is_zero()) & bool((m[i,3]-
m[i+1+j,3]).is_zero()) & bool((m[i,4]-m[i+1+j,4]).is_zero()):
                        m[i,0]=m[i,0]+m[i+1+j,0]
                        m[i+1+j,0]=0
                        m[i+1+j,1]=0
                        n=n+1
                    else:
                        n=n
        if bool(n==0):
            return m
        else:
            return delete_zeroes(m)

def matrix_full_simplify(v):
    m=matrix(SR,v.nrows(),v.ncols())
    n=0
    for i in range(v.nrows()):
        if bool(v[i,0].is_zero()):
            n=n+1
        else:
            for j in range(v.ncols()):
               m[i-n,j]=v[i,j]
    m=m.submatrix(0,0,v.nrows()-n,v.ncols())
    if bool(m.ncols()==4):
        n=0
        mlength=m.nrows()
        for i in range(mlength):
            if bool(m[i,0].is_zero()):
                n=n
            else:
                for j in range(mlength-i-1):
                    if bool((m[i,1]-m[i+1+j,1]).is_zero()) &
bool((m[i,2]-m[i+1+j,2]).is_zero()) & bool((m[i,3]-
m[i+1+j,3]).is_zero()):
                        m[i,0]=m[i,0]+m[i+1+j,0]
                        m[i+1+j,0]=0
```

```
                        n=n+1
                    else:
                        n=n
                m[i,0]=m[i,0].full_simplify()
        if bool(n==0):
            return m
        else:
            return delete_zeroes(m)
    else:
        n=0
        mlentgh=m.nrows()
        for i in range(mlentgh):
            if bool(m[i,0].is_zero()):
                n=n
            else:
                for j in range(mlentgh-i-1):
                    if bool((m[i,1]-m[i+1+j,1]).is_zero()) &
bool((m[i,2]-m[i+1+j,2]).is_zero()) & bool((m[i,3]-
m[i+1+j,3]).is_zero()) & bool((m[i,4]-m[i+1+j,4]).is_zero()):
                        m[i,0]=m[i,0]+m[i+1+j,0]
                        m[i+1+j,0]=0
                        n=n+1
                    else:
                        n=n
                m[i,0]=m[i,0].full_simplify()
        if bool(n==0):
            return m
        else:
            return delete_zeroes(m)


def real_part(v):
    m=matrix(SR,2*v.nrows(),v.ncols())
    if v.ncols()==4:
        for i in range(v.nrows()):
            m[2*i,0]=v[i,0]/2
            m[2*i,1]=v[i,1]
            m[2*i,2]=v[i,2]
            m[2*i,3]=v[i,3]
            m[2*i+1,0]=conjugate(v[i,0])/2
            m[2*i+1,1]=v[i,2]
            m[2*i+1,2]=v[i,1]
            m[2*i+1,3]=v[i,3]
        n=0
        mlentgh=m.nrows()
        for i in range(mlentgh):
            if m[i,0].is_zero():
                n=n
            else:
                for j in range(mlentgh-i-1):
                    if bool(m[i,1]==m[i+1+j,1]) &
bool(m[i,2]==m[i+1+j,2]) & bool(m[i,3]==m[i+1+j,3]):
                        m[i,0]=m[i,0]+m[i+1+j,0]
                        m[i+1+j,0]=0                #We delete the
```

```
line we have just added
                        n=n+1
                    else:
                        n=n
    else:
        for i in range(v.nrows()):
            m[2*i,0]=v[i,0]/2
            m[2*i,1]=v[i,1]
            m[2*i,2]=v[i,2]
            m[2*i,3]=v[i,3]
            m[2*i,4]=v[i,4]
            m[2*i+1,0]=conjugate(v[i,0])/2
            m[2*i+1,1]=conjugate(v[i,1])
            m[2*i+1,2]=v[i,3]
            m[2*i+1,3]=v[i,2]
            m[2*i+1,4]=v[i,4]
        n=0
        mlength=m.nrows()
        for i in range(mlength):
            if m[i,0].is_zero():
               n=n
            else:
               for j in range(mlength-i-1):
                   if bool(m[i,1]==m[i+1+j,1]) &
bool(m[i,2]==m[i+1+j,2]) & bool(m[i,3]==m[i+1+j,3]) &
bool(m[i,4]==m[i+1+j,4]):
                       m[i,0]=m[i,0]+m[i+1+j,0]
                       m[i+1+j,0]=0                #We delete the
line we have just added
                       n=n+1
                   else:
                       n=n
    return delete_zeroes(m)

def real_part2(v):
    n=v.nrows()
    m=matrix(SR,2*v.nrows(),v.ncols())
    if v.ncols()==4:
        for i in range(v.nrows()):
            m[i,0]=v[i,0]/2
            m[i,1]=v[i,1]
            m[i,2]=v[i,2]
            m[i,3]=v[i,3]
            m[i+n,0]=conjugate(v[i,0])/2
            m[i+n,1]=v[i,2]
            m[i+n,2]=v[i,1]
            m[i+n,3]=v[i,3]
        n=0
        mlentgh=m.nrows()
        for i in range(mlentgh):
            if m[i,0].is_zero():
                n=n
            else:
                for j in range(mlentgh-i-1):
```

```
                    if bool(m[i,1]==m[i+1+j,1]) &
bool(m[i,2]==m[i+1+j,2]) & bool(m[i,3]==m[i+1+j,3]):
                        m[i,0]=m[i,0]+m[i+1+j,0]
                        m[i+1+j,0]=0                #We delete the
line we have just added
                        n=n+1
                    else:
                        n=n
    else:
        for i in range(v.nrows()):
            m[i,0]=v[i,0]/2
            m[i,1]=v[i,1]
            m[i,2]=v[i,2]
            m[i,3]=v[i,3]
            m[i,4]=v[i,4]
            m[i+n,0]=conjugate(v[i,0])/2
            m[i+n,1]=conjugate(v[i,1])
            m[i+n,2]=v[i,3]
            m[i+n,3]=v[i,2]
            m[i+n,4]=v[i,4]
        n=0
        mlength=m.nrows()
        for i in range(mlength):
            if m[i,0].is_zero():
               n=n
            else:
               for j in range(mlength-i-1):
                   if bool(m[i,1]==m[i+1+j,1]) &
bool(m[i,2]==m[i+1+j,2]) & bool(m[i,3]==m[i+1+j,3]) &
bool(m[i,4]==m[i+1+j,4]):
                       m[i,0]=m[i,0]+m[i+1+j,0]
                       m[i+1+j,0]=0                #We delete the
line we have just added
                       n=n+1
                   else:
                       n=n
    return delete_zeroes(m)


def scal(v,w): #returns the scalar product of two vectors
    l = v.nrows()*w.nrows()
    m = matrix(SR,l,4)
    if v.ncols()==4:
        for i in range(v.nrows()):
            for j in range(w.nrows()):
                m[i*w.nrows()+j,0]=v[i,0]*w[j,0]
                m[i*w.nrows()+j,1]=v[i,1]+w[j,1]
                m[i*w.nrows()+j,2]=v[i,2]+w[j,2]
                m[i*w.nrows()+j,3]=v[i,3]+w[j,3]
        return m
    else:
        for i in range(v.nrows()):
           for j in range(w.nrows()):
                m[i*w.nrows()+j,0]=v[i,0]*v[i,1]*w[j,0]*w[j,1]
```

```
                m[i*w.nrows()+j,1]=v[i,2]+w[j,2]
                m[i*w.nrows()+j,2]=v[i,3]+w[j,3]
                m[i*w.nrows()+j,3]=v[i,4]+w[j,4]
        return m


def matrix_sort(v): # the end will be used repetedly to obtain easy
reading code
    m=matrix(SR,v.nrows(),v.ncols())
    if v.ncols()==4:
        for i in range(v.nrows()):
            m[i,0]=v[i,2]
            m[i,1]=v[i,3]
            m[i,2]=v[i,0]
            m[i,3]=v[i,1]
        m=matrix(sorted(m))
        for i in range(v.nrows()):
            temp0=m[i,2]
            temp1=m[i,3]
            m[i,2]=m[i,0]
            m[i,3]=m[i,1]
            m[i,0]=temp0
            m[i,1]=temp1
        return m
    else:
        for i in range(v.nrows()):
            m[i,0]=v[i,2]
            m[i,1]=v[i,3]
            m[i,2]=v[i,4]
            m[i,3]=v[i,0]
            m[i,4]=v[i,1]
        m=matrix(sorted(m))
        for i in range(v.nrows()):
            temp0=m[i,0]
            temp1=m[i,1]
            m[i,0]=m[i,3]
            m[i,1]=m[i,4]
            m[i,4]=m[i,2]
            m[i,2]=temp0
            m[i,3]=temp1
        return m

def prod(scalar,vector): #returns the product of a scalar with a
vector
    l= scalar.nrows()*vector.nrows()
    m=matrix(SR,l,5)
    for i in range(scalar.nrows()):
        for j in range(vector.nrows()):
            m[i*vector.nrows()+j,0]=scalar[i,0]*vector[j,0]
            m[i*vector.nrows()+j,1]=vector[j,1]
            m[i*vector.nrows()+j,2]=scalar[i,1]+vector[j,2]
            m[i*vector.nrows()+j,3]=scalar[i,2]+vector[j,3]
            m[i*vector.nrows()+j,4]=scalar[i,3]+vector[j,4]
    return m
```

```
def bar(v):
    m=matrix(SR,v.nrows(),v.ncols())
    if v.ncols()==4:
        for i in range(v.nrows()):
            m[i,0]=conjugate(v[i,0])
            m[i,1]=v[i,2]
            m[i,2]=v[i,1]
            m[i,3]=v[i,3]
        return m
    else:
        for i in range(v.nrows()):
            m[i,0]=conjugate(v[i,0])
            m[i,1]=conjugate(v[i,1])
            m[i,2]=v[i,3]
            m[i,3]=v[i,2]
            m[i,4]=v[i,4]
        return m

def intz(v):
    length=0
    for i in range(v.nrows()):
        if bool((v[i,v.ncols()-3]+1).is_zero()): # if the
coefficient is z^{-1}\z^{b}\log^p|z|, the primitive has only one
components
            length=length+1
        else:
            length=length+v[i,v.ncols()-1]+1 # if the coefficient is
z^{a}\z^{b}\log^p|z| with a\neq -1, then the primitive has p+1
components
    m=matrix(SR,length,v.ncols())
    n=0
    if v.ncols()==4:
        for i in range(v.nrows()):
            if (v[i,1]+1).is_zero():        #integration of 1/z
                m[i+n,0]=2*v[i,0]/(v[i,3]+1)
                m[i+n,1]=0
                m[i+n,2]=v[i,2]
                m[i+n,3]=v[i,3]+1
            else:
                if v[i,3].is_zero():
                    m[i+n,0]=v[i,0]/(v[i,1]+1)
                    m[i+n,1]=v[i,1]+1
                    m[i+n,2]=v[i,2]
                else:
                    a0=1/(v[i,1]+1)
                    m[i+n,0]=a0*v[i,0]
                    m[i+n,1]=v[i,1]+1
                    m[i+n,2]=v[i,2]
                    m[i+n,3]=v[i,3]
                    n=n+1
                    for j in range(v[i,3]):
                        a0=-(v[i,3]-j)/(2*(v[i,1]+1))*a0
                        m[i+n,0]=a0*v[i,0]
```

```
                        m[i+n,1]=v[i,1]+1
                        m[i+n,2]=v[i,2]
                        m[i+n,3]=v[i,3]-j-1
                        n=n+1
                    n=n-1
        return m
    else:
        for i in range(v.nrows()):
            if (v[i,2]+1).is_zero():        #integration of 1/z
                m[i+n,0]=2*v[i,0]/(v[i,4]+1)
                m[i+n,1]=v[i,1]
                m[i+n,2]=0
                m[i+n,3]=v[i,3]
                m[i+n,4]=v[i,4]+1
            else:
                if v[i,4].is_zero():
                    m[i+n,0]=v[i,0]/(v[i,2]+1)
                    m[i+n,1]=v[i,1]
                    m[i+n,2]=v[i,2]+1
                    m[i+n,3]=v[i,3]
                else:
                    a0=1/(v[i,2]+1)
                    m[i+n,0]=a0*v[i,0]
                    m[i+n,1]=v[i,1]
                    m[i+n,2]=v[i,2]+1
                    m[i+n,3]=v[i,3]
                    m[i+n,4]=v[i,4]
                    n=n+1
                    for j in range(v[i,4]):
                        a0=-(v[i,4]-j)/(2*(v[i,2]+1))*a0
                        m[i+n,0]=a0*v[i,0]
                        m[i+n,1]=v[i,1]
                        m[i+n,2]=v[i,2]+1
                        m[i+n,3]=v[i,3]
                        m[i+n,4]=v[i,4]-j-1
                        n=n+1
                    n=n-1
        return m

def intzb(v):
    length=0
    for i in range(v.nrows()):
        if bool((v[i,v.ncols()-2]+1).is_zero()): # if the
coefficient is z^{-1}\z^{b}\log^p|z|, the primitive has only one
components
            length=length+1
        else:
            length=length+v[i,v.ncols()-1]+1 # if the coefficient is
z^{a}\z^{b}\log^p|z| with a\neq -1, then the primitive has p+1
components
    m=matrix(SR,length,v.ncols())
    n=0
    if v.ncols()==4:
        for i in range(v.nrows()):
```

```
        if (v[i,2]+1).is_zero():        #integration of 1/z
            m[i+n,0]=2*v[i,0]/(v[i,3]+1)
            m[i+n,1]=v[i,1]
            m[i+n,2]=0
            m[i+n,3]=v[i,3]+1
        else:
            if v[i,3].is_zero():
                m[i+n,0]=v[i,0]/(v[i,1]+1)
                m[i+n,1]=v[i,1]
                m[i+n,2]=v[i,2]+1
            else:
                a0=1/(v[i,2]+1)
                m[i+n,0]=a0*v[i,0]
                m[i+n,1]=v[i,1]
                m[i+n,2]=v[i,2]+1
                m[i+n,3]=v[i,3]
                n=n+1
                for j in range(v[i,3]):
                    a0=-(v[i,3]-j)/(2*(v[i,2]+1))*a0
                    m[i+n,0]=a0*v[i,0]
                    m[i+n,1]=v[i,1]
                    m[i+n,2]=v[i,2]+1
                    m[i+n,3]=v[i,3]-j-1
                    n=n+1
                n=n-1
    return m
else:
    for i in range(v.nrows()):
        if (v[i,3]+1).is_zero():        #integration of 1/z
            m[i+n,0]=2*v[i,0]/(v[i,4]+1)
            m[i+n,1]=v[i,1]
            m[i+n,2]=v[i,2]
            m[i+n,3]=0
            m[i+n,4]=v[i,4]+1
        else:
            if v[i,4].is_zero():
                m[i+n,0]=v[i,0]/(v[i,3]+1)
                m[i+n,1]=v[i,1]
                m[i+n,2]=v[i,2]
                m[i+n,3]=v[i,3]+1
            else:
                a0=1/(v[i,3]+1)
                m[i+n,0]=a0*v[i,0]
                m[i+n,1]=v[i,1]
                m[i+n,2]=v[i,2]
                m[i+n,3]=v[i,3]+1
                m[i+n,4]=v[i,4]
                n=n+1
                for j in range(v[i,4]):
                    a0=-(v[i,4]-j)/(2*(v[i,3]+1))*a0
                    m[i+n,0]=a0*v[i,0]
                    m[i+n,1]=v[i,1]
                    m[i+n,2]=v[i,2]
                    m[i+n,3]=v[i,3]+1
```

```
                        m[i+n,4]=v[i,4]-j-1
                        n=n+1
                    n=n-1
        return m

def diffz(v):
    length=v.nrows()
    if v.ncols()==4:
        for i in range(v.nrows()):
            if v[i,3]==0:
                length=length
            else:
                length=length+1
    else:
        for i in range(v.nrows()):
            if v[i,4]==0:
                length=length
            else:
                length=length+1
    m=matrix(SR,length,v.ncols())
    n=0
    if v.ncols()==4:
        for i in range(v.nrows()):
            if v[i,0].is_zero():
                n=n+1
            else:
                if bool(v[i,1]==0) & bool(v[i,3]==0):
                    n=n+1
                else:
                    if v[i,3]==0:
                        m[i-n,0]=v[i,1]*v[i,0]
                        m[i-n,1]=v[i,1]-1
                        m[i-n,2]=v[i,2]
                    else:
                        if v[i,1]==0:
                            m[i-n,0]=v[i,3]*v[i,0]/2   # the factor
1/2 comes from the derivation of the log
                            m[i-n,1]=v[i,1]-1
                            m[i-n,2]=v[i,2]
                            m[i-n,3]=v[i,3]-1
                        else:
                            m[i-n,0]=v[i,1]*v[i,0]
                            m[i-n,1]=v[i,1]-1
                            m[i-n,2]=v[i,2]
                            m[i-n,3]=v[i,3]
                            m[i-n+1,0]=v[i,3]*v[i,0]/2
                            m[i-n+1,1]=v[i,1]-1
                            m[i-n+1,2]=v[i,2]
                            m[i-n+1,3]=v[i,3]-1
                            n=n-1            # a we added another
term, we need to shift lines
    else:
        for i in range(v.nrows()):
            if v[i,0].is_zero():
```

```
                n=n+1
            else:
                if bool(v[i,2]==0) & bool(v[i,4]==0):
                    n=n+1
                else:
                    if v[i,4]==0:
                        m[i-n,0]=v[i,2]*v[i,0]
                        m[i-n,1]=v[i,1]
                        m[i-n,2]=v[i,2]-1
                        m[i-n,3]=v[i,3]
                    else:
                        if v[i,2]==0:
                            m[i-n,0]=v[i,4]*v[i,0]/2   # the factor
1/2 comes from the derivation of the log
                            m[i-n,1]=v[i,1]
                            m[i-n,2]=v[i,2]-1
                            m[i-n,3]=v[i,3]
                            m[i-n,4]=v[i,4]-1
                        else:
                            m[i-n,0]=v[i,2]*v[i,0]
                            m[i-n,1]=v[i,1]
                            m[i-n,2]=v[i,2]-1
                            m[i-n,3]=v[i,3]
                            m[i-n,4]=v[i,4]
                            m[i-n+1,0]=v[i,4]*v[i,0]/2
                            m[i-n+1,1]=v[i,1]
                            m[i-n+1,2]=v[i,2]-1
                            m[i-n+1,3]=v[i,3]
                            m[i-n+1,4]=v[i,4]-1
                            n=n-1              # we added another
term, we need to shift lines
    return m.submatrix(0,0,v.nrows()-n,v.ncols())

def diffzb(v):
    length=v.nrows()
    if v.ncols()==4:
        for i in range(v.nrows()):
            if v[i,3]==0:
                length=length
            else:
                length=length+1
    else:
        for i in range(v.nrows()):
            if v[i,4]==0:
                length=length
            else:
                length=length+1
    m=matrix(SR,length,v.ncols())
    n=0
    if v.ncols()==4:
        for i in range(v.nrows()):
            if v[i,0].is_zero():
                n=n+1
            else:
```

```
                if bool(v[i,2]==0) & bool(v[i,3]==0):
                    n=n+1
                else:
                    if v[i,3]==0:
                        m[i-n,0]=v[i,2]*v[i,0]
                        m[i-n,1]=v[i,1]
                        m[i-n,2]=v[i,2]-1
                    else:
                        if v[i,1]==0:
                            m[i-n,0]=v[i,3]*v[i,0]/2   # the factor
1/2 comes from the derivation of the log
                            m[i-n,1]=v[i,1]
                            m[i-n,2]=v[i,2]-1
                            m[i-n,3]=v[i,3]-1
                        else:
                            m[i-n,0]=v[i,2]*v[i,0]
                            m[i-n,1]=v[i,1]
                            m[i-n,2]=v[i,2]-1
                            m[i-n,3]=v[i,3]
                            m[i-n+1,0]=v[i,3]*v[i,0]/2
                            m[i-n+1,1]=v[i,1]
                            m[i-n+1,2]=v[i,2]-1
                            m[i-n+1,3]=v[i,3]-1
                            n=n-1            # we added another
term, we need to shift lines
    else:
        for i in range(v.nrows()):
            if v[i,0].is_zero():
                n=n+1
            else:
                if bool(v[i,3]==0) & bool(v[i,4]==0):
                    n=n+1
                else:
                    if v[i,4]==0:
                        m[i-n,0]=v[i,3]*v[i,0]
                        m[i-n,1]=v[i,1]
                        m[i-n,2]=v[i,2]
                        m[i-n,3]=v[i,3]-1
                    else:
                        if v[i,2]==0:
                            m[i-n,0]=v[i,4]*v[i,0]/2   # the factor
1/2 comes from the derivation of the log
                            m[i-n,1]=v[i,1]
                            m[i-n,2]=v[i,2]
                            m[i-n,3]=v[i,3]-1
                            m[i-n,4]=v[i,4]-1
                        else:
                            m[i-n,0]=v[i,3]*v[i,0]
                            m[i-n,1]=v[i,1]
                            m[i-n,2]=v[i,2]
                            m[i-n,3]=v[i,3]-1
                            m[i-n,4]=v[i,4]
                            m[i-n+1,0]=v[i,4]*v[i,0]/2
                            m[i-n+1,1]=v[i,1]
```

```
                            m[i-n+1,2]=v[i,2]
                            m[i-n+1,3]=v[i,3]-1
                            m[i-n+1,4]=v[i,4]-1
                            n=n-1             # we added another
term, we need to shift lines
    return m.submatrix(0,0,v.nrows()-n,v.ncols())


def throw(v,bound): #THis algorithm permits to throw out all errors
larger or equal than bound
    m=matrix(SR,v.nrows(),v.ncols())
    n=0
    if v.ncols()==4:
        for i in range(v.nrows()):
            if v[i,1]+v[i,2]<bound:
                m[i-n,0]=v[i,0]
                m[i-n,1]=v[i,1]
                m[i-n,2]=v[i,2]
                m[i-n,3]=v[i,3]    #The logarithm do not matter for
throw, as we look at power of z up to |z|^{-\epsilon}
            else:
                n=n+1
    else:
        for i in range(v.nrows()):
            if v[i,2]+v[i,3]<bound:
                m[i-n,0]=v[i,0]
                m[i-n,1]=v[i,1]
                m[i-n,2]=v[i,2]
                m[i-n,3]=v[i,3]
                m[i-n,4]=v[i,4]
            else:
                n=n+1
    return m.submatrix(0,0,v.nrows()-n,v.ncols())




def mult_scalar(l,v):
    m=matrix(SR,v.nrows(),v.ncols())
    for i in range(v.nrows()):
        m[i,0]=l*v[i,0]
        for j in range(v.ncols()-1):
            m[i,j+1]=v[i,j+1]
    return m

def sum_matrix(v,w):
    d=v.nrows()+w.nrows()
    d1=v.nrows()
    m=matrix(SR,d,v.ncols())
    for i in range(v.nrows()):
        for j in range(v.ncols()):
            m[i,j]=v[i,j]
    for i in range(w.nrows()):
        for j in range(w.ncols()):
```

```
            m[i+d1,j]=w[i,j]
    return factor_simplify(m)

def sing_quartic(ginv,h0,bound):
    mz=diffz(h0)
    mzzb=diffzb(mz)
    mzb=diffzb(h0)
    h1=scal(mzzb,h0)
    h2=scal(mz,mzb)
    return
factor_simplify(throw(scal(ginv,sum_matrix(h1,mult_scalar(-1,h2))),b
ound))

def intQ(dphi,ginv,H,h0,bound):
    m1=throw(mult_scalar(-1,prod(scal(H,H),dphi)),bound-1)

m2=throw(mult_scalar(-2,prod(scal(H,h0),prod(ginv,bar(dphi)))),bound
-1)
    Q=intz(sum_matrix(m1,m2))
    return matrix_full_simplify(Q)

def power_divide(v,nz,nzb):
    m=matrix(SR,v.nrows(),v.ncols())
    for i in range(v.nrows()):
        for j in range(v.ncols()):
            if j==v.ncols()-3:
                m[i,j]=v[i,j]-nz
            else:
                if j==v.ncols()-2:
                    m[i,j]=v[i,j]-nzb
                else:
                    m[i,j]=v[i,j]
    return m

def simplify_matrix(v):
    m=matrix(SR,v.nrows(),v.ncols())
    for i in range(v.nrows()):
        for j in range(v.ncols()):
            if j==0:
                m[i,j]=v[i,j].full_simplify()
            else:
                m[i,j]=v[i,j]
    return m

def ellipse(v,n):
    m=matrix(SR,v.nrows(),v.ncols())
    for i in range(v.nrows()):
        for j in range(v.ncols()):
            if bool(j==v.ncols()-1) & bool(v[i,j]==n-1):
                m[i,0]=0
            else:
                m[i,j]=v[i,j]
    return delete_zeroes(m)
```

```
def taylor_inverse(v,order,bound):            # gives the Taylor
development up to order inverse of                  a function g
given as g(x)=1+f(x)
    m=matrix(SR,v.nrows()-1,v.ncols())
    for i in range(v.nrows()-1):
        for j in range(v.ncols()):
            if bool(j==0):
                m[i,j]=-v[i+1,j]
            else:
                m[i,j]=v[i+1,j]
    vbis=m
    temp=m
    for i in range(order-1):
        temp=scal(temp,vbis)
        m=sum_matrix(m,mult_scalar((-1)^i,throw(temp,bound)))
    add1=matrix(SR,1,v.ncols())
    add1[0,0]=1
    return factor_simplify(sum_matrix(add1,m))

def search_coefficient(h0,a,b):
    m=matrix(SR,1,1)
    for i in range(h0.nrows()):
        for j in range(h0.nrows()-i-1):
            if bool((h0[i,2]+h0[i+j+1,2]-a).is_zero()) &
bool((h0[i,3]+h0[i+j+1,3]-b).is_zero()):

m[0,0]=m[0,0]+h0[i,0]*h0[i,1]*h0[i+j+1,0]*h0[i+j+1,1]*(h0[i,2]-
h0[i+j+1,2])*(h0[i,3]-h0[i+j+1,3])
            else:
                m=m
    return m[0,0].full_simplify()

def search_powers(v,theta_0,bound): # with this algorithm, we obtain
all possible powers arising in the Taylor expansion of the quartic
form but delete all holomorphic coefficients
    l=v.nrows()
    c=v.ncols()
    m=matrix(SR,l*(l-1)/2,2)
    n=0
    for i in range(v.nrows()):
        for j in range(v.nrows()-i-1):
            if bool((v[i,c-2]-v[i+j+1,c-2]).is_zero()) or
bool((v[i,c-1]-v[i+j+1,c-1]).is_zero()) or
bool((v[i,c-2]+v[i+j+1,c-2]+v[i,c-1]+v[i+j+1,c-1]-2*theta_0)>bound-1
): #We check if the product is trivial or not
                n=n
            else:
                m[n,0]=v[i,c-2]+v[i+j+1,c-2]-theta_0
                m[n,1]=v[i,c-1]+v[i+j+1,c-1]-theta_0
                n=n+1
    for i in range(m.nrows()):         # We delete multiple
occurrences
            if bool(m[i,1].is_zero()): # avoids making useless
tests, as the holomorphic coefficients will be deleted by
```

```
delete_hol()
                n=n
            else:
                for j in range(m.nrows()-i-1):
                    if bool((m[i,0]-m[i+j+1,0]).is_zero()) &
bool((m[i,1]-m[i+j+1,1]).is_zero()):
                        m[i+j+1,1]=0
                    else:
                        n=n
    return delete_hol(m)

def Gauss_curvature(e2u,e2uinv,bound):
    temp1=throw(scal(e2uinv,diffz(diffzb(e2u))),bound)
    temp2=throw(scal(e2uinv,scal(diffz(e2u),diffzb(e2u))),bound)
    return
factor_simplify(mult_scalar(2,sum_matrix(temp1,mult_scalar(-1,temp2)
)))

def Weingarten(dzphi,g,ginv,bound1,bound2):
    temp1=diffz(dzphi)
    dzlambda=throw(scal(ginv,diffz(g)),bound1)
    temp2=throw(mult_scalar(-1,prod(dzlambda,dzphi)),bound2)
    return
matrix_full_simplify(mult_scalar(2,sum_matrix(temp1,temp2)))

def Weingarten_simpler(dzphi,g,ginv,bound1,bound2):
    temp1=diffz(dzphi)
    dzlambda=throw(scal(ginv,diffz(g)),bound1)
    temp2=throw(mult_scalar(-1,prod(dzlambda,dzphi)),bound2)
    return
matrix_full_simplify(mult_scalar(2,sum_matrix(temp1,temp2)))


def Gauss_Weingarten(e2u,e2uinv,ginv,h0,bound):
    temp1=throw(scal(e2uinv,diffz(diffzb(e2u))),bound)

temp2=throw(scal(scal(e2uinv,e2uinv),scal(diffz(e2u),diffzb(e2u))),b
ound)
    K=mult_scalar(2,sum_matrix(temp1,mult_scalar(-1,temp2)))
    temp3=throw(scal(ginv,scal(h0,h0)),bound)
    return factor_simplify(throw(scal(K,temp3),bound))


n=3


from sage.misc.latex import MathJax
mj = MathJax()

dzphi=matrix([[1,a0,n-1,0,0],[1,a1,n,0,0],[1,a2,n+1,0,0],
[1/12,c1,1,3,0],[1/8,c1b,2,2,0]])

H=real_part(matrix([1,c1,2-n,0,0]))
```

```
g=matrix([[1,n-1,n-1,0],[2*abs(a1)^2,n,n,0],[alpha0,n,n-1,0],
[alpha0b,n-1,n,0],[alpha1,n+1,n-1,0],[alpha1b,n-1,n+1,0]])

ginv=power_divide(taylor_inverse(power_divide(g,n-1,n-1),2,3),n-1,n-
1)

h0=Weingarten(dzphi,g,ginv,2,n+1)

latex(h0)

Q=intQ(dzphi,ginv,H,h0,4-n)

Htemp=matrix_full_simplify(real_part(Q))

latex(H)

mj(dzphi),mj(g),mj(H),mj(h0),mj(Htemp)

H2=real_part2(matrix([[1,c1,2-n,0,0],[1,b1,-1,1,0],[1,c2,0,0,0],
[1,gamma1,0,0,1]]))

latex(H2)

dzphi2=sum_matrix(matrix([[1,a0,n-1,0,0],[1,a1,n,0,0],
[1,a2,n+1,0,0],
[1,a3,n+2,0,0]]),mult_scalar(1/2,matrix_sort(matrix_full_simplify(in
tzb(throw(prod(g,H2),n+2))))))

latex(dzphi2)

conf=matrix_full_simplify(throw(scal(dzphi2,dzphi2),2*n+2))

g2temp=matrix_full_simplify(mult_scalar(2,throw(scal(dzphi2,bar(dzph
i2)),2*n+2)))

latex(conf),latex(g2temp)

g2=sum_matrix(matrix([[1,n-1,n-1,0],
[2*abs(a1)^2,n,n,0]]),real_part2(matrix([[2*alpha0,n,n-1,0],
[2*alpha1,n+1,n-1,0],[2*alpha2,1,2*n,0],[2*alpha3,5,2,0],
[2*alpha4,4,3,0]])))

latex(g2)

g2inv=power_divide(taylor_inverse(power_divide(g2,n-1,n-1),4,5),n-1,
n-1)

h02=Weingarten(dzphi2,g2,g2inv,3,n+2)

latex(h02)


phi=sum_matrix(matrix([[1/n,a0b,0,n,0],[1/(n+1),a1b,0,n+1,0],[1/
(n+2),a2b,0,n+2,0],[1/(n+3),a3b,0,n+3,0]]),intz(dzphi2))
```

```
H5m1=H2

H5=H2

m1=throw(prod(scal(H2,H2),throw(dzphi2,n)),4-n)

h02m2=throw(h02,n+1)

H5m2=throw(H5,4-n)

g2invm2=throw(g2inv,4-2*n)

dzphi2m2=throw(dzphi2,n+1)

m2=throw(mult_scalar(2,prod(scal(H5m2,h02m2),prod(g2invm2,bar(dzphi2
m2)))),4-n)

alpha=factor_simplify(sum_matrix(diffz(H5),sum_matrix(m1,m2)))

latex(alpha)



phitemp=throw(phi,n+2)

temp1=factor_simplify(throw(prod(throw(scal(phitemp,phitemp),2*n+2),
alpha),n+3))

temp2=mult_scalar(2,factor_simplify(throw(prod(throw(scal(phitemp,al
pha),2*n+2),phitemp),n+3)))

phitemp2=throw(phi,n+3)

tempfate=throw(scal(throw(g2inv,5-2*n),diffzb(scal(phitemp2,phitemp2
))),4)

fate3=throw(prod(tempfate,h02),n+3)

fate=throw(scal(h02,phi),2*n+2)

fatebis=throw(prod(throw(g2inv,5-2*n),throw(bar(dzphi2),n+2)),4-n)

fate2=matrix_full_simplify(throw(prod(fate,fatebis),n+3))

end1=sum_matrix(temp1,mult_scalar(-1,temp2))

end2=sum_matrix(mult_scalar(2,fate2),mult_scalar(-1,fate3))

endend=matrix_sort(matrix_full_simplify(real_part(diffzb(sum_matrix(
end1,end2)))))

latex(endend)
```

```
///
\left(\begin{array}{rrrrr}
\frac{1}{6} \, \overline{A_{0}}^{2} & C_{1} & -2 & 5 & 0 \\
\frac{5}{72} \, \overline{A_{0}}^{2} & B_{1} & -2 & 6 & 0 \\
-\frac{1}{24} \, C_{1} \overline{A_{0}} & \overline{A_{1}} & -2 & 6
& 0 \\
-\frac{1}{24} \, \overline{A_{0}}^{2} \overline{\alpha_{0}} +
\frac{1}{3} \, \overline{A_{0}} \overline{A_{1}} & C_{1} & -2 & 6 &
0 \\
-2 \, \alpha_{2} \overline{A_{0}}^{2} + \frac{1}{9} \,
{\left(\overline{A_{0}}^{3} \overline{\alpha_{0}} - \overline{A_{0}}
^{2} \overline{A_{1}}\right)} C_{1} & A_{0} & -2 & 6 & 0 \\
2 \, A_{0} \alpha_{2} \overline{A_{0}} - \frac{1}{72} \, {\left(16
\, A_{0} \overline{A_{0}}^{2} \overline{\alpha_{0}} - 16 \, A_{0}
\overline{A_{0}} \overline{A_{1}} - 3 \, \overline{A_{0}}
\overline{\alpha_{0}}\right)} C_{1} + \frac{11}{72} \, B_{1}
\overline{A_{0}} & \overline{A_{0}} & -2 & 6 & 0 \\
-\frac{1}{6} \, \overline{A_{0}}^{2} & \gamma_{1} & -1 & 5 & 0 \\
16 \, \overline{A_{0}}^{2} \overline{\alpha_{0}}^{3} & A_{0} & -1 &
5 & 0 \\
-16 \, A_{0} \overline{A_{0}} \overline{\alpha_{0}}^{3} + \frac{2}
{3} \, {\left(A_{0} \alpha_{0} \overline{A_{0}}^{2} - A_{1}
\overline{A_{0}}^{2}\right)} C_{1} & \overline{A_{0}} & -1 & 5 & 0 \
\\
-2 \, \overline{A_{0}}^{2} & A_{1} & 0 & 2 & 0 \\
2 \, \alpha_{0} \overline{A_{0}}^{2} & A_{0} & 0 & 2 & 0 \\
-2 \, A_{0} \alpha_{0} \overline{A_{0}} + 2 \, A_{1}
\overline{A_{0}} & \overline{A_{0}} & 0 & 2 & 0 \\
2 \, \overline{A_{0}}^{2} \overline{\alpha_{0}} - 4 \,
\overline{A_{0}} \overline{A_{1}} & A_{1} & 0 & 3 & 0 \\
-2 \, A_{0} \alpha_{0} \overline{A_{0}} + 2 \, A_{1}
\overline{A_{0}} & \overline{A_{1}} & 0 & 3 & 0 \\
4 \, {\left| A_{1} \right|}^{2} \overline{A_{0}}^{2} - 4 \,
\alpha_{0} \overline{A_{0}}^{2} \overline{\alpha_{0}} + 4 \,
\alpha_{0} \overline{A_{0}} \overline{A_{1}} & A_{0} & 0 & 3 & 0 \\
-4 \, A_{0} {\left| A_{1} \right|}^{2} \overline{A_{0}} + 4 \, A_{0}
\alpha_{0} \overline{A_{0}} \overline{\alpha_{0}} - 2 \, A_{1}
\overline{A_{0}} \overline{\alpha_{0}} - 2 \, {\left(A_{0}
\alpha_{0} - A_{1}\right)} \overline{A_{1}} & \overline{A_{0}} & 0 &
3 & 0 \\
-\frac{11}{36} \, \overline{A_{0}}^{2} & \overline{B_{1}} & 0 & 4 &
0 \\
-2 \, {\left(\overline{\alpha_{0}}^{2} - \overline{\alpha_{1}}
\right)} \overline{A_{0}}^{2} + 4 \, \overline{A_{0}}
\overline{A_{1}} \overline{\alpha_{0}} - 2 \, \overline{A_{1}}^{2} -
4 \, \overline{A_{0}} \overline{A_{2}} & A_{1} & 0 & 4 & 0 \\
-2 \, A_{0} \alpha_{0} \overline{A_{0}} + 2 \, A_{1}
\overline{A_{0}} & \overline{A_{2}} & 0 & 4 & 0 \\
4 \, {\left(2 \, A_{0} \overline{A_{0}} \overline{\alpha_{0}} -
A_{0} \overline{A_{1}}\right)} {\left| A_{1} \right|}^{2} - 2 \,
{\left(19 \, \alpha_{0} \overline{\alpha_{0}}^{2} - 2 \, \alpha_{0}
\overline{\alpha_{1}} + \overline{\alpha_{4}}\right)} A_{0}
\overline{A_{0}} + 2 \, {\left(\overline{\alpha_{0}}^{2} -
\overline{\alpha_{1}}\right)} A_{1} \overline{A_{0}} + 2 \, {\left(2
```

$$
\begin{array}{cccccc}
\, A_{0} \alpha_{0} \overline{\alpha_{0}} - A_{1} \overline{\alpha_{0}}\right)} \overline{A_{1}} - 2 \, {\left(A_{0} \alpha_{0} - A_{1}\right)} \overline{A_{2}} + \frac{7}{36} \, \overline{A_{0}} \overline{B_{1}} + \frac{5}{9} \, {\left(A_{0} \alpha_{0} \overline{A_{0}}^{2} - A_{1} \overline{A_{0}}^{2}\right)} \overline{C_{1}} & \overline{A_{0}} & 0 & 4 & 0 \\
-8 \, \alpha_{0} \overline{A_{0}} \overline{A_{1}} \overline{\alpha_{0}} - 8 \, {\left(\overline{A_{0}}^{2} \overline{\alpha_{0}} - \overline{A_{0}} \overline{A_{1}}\right)} {\left| A_{1} \right|}^{2} + 2 \, {\left(19 \, \alpha_{0} \overline{\alpha_{0}}^{2} - 2 \, \alpha_{0} \overline{\alpha_{1}} + \overline{\alpha_{4}}\right)} \overline{A_{0}}^{2} + 2 \, \alpha_{0} \overline{A_{1}}^{2} + 4 \, \alpha_{0} \overline{A_{0}} \overline{A_{2}} & A_{0} & 0 & 4 & 0 \\
-4 \, A_{0} {\left| A_{1} \right|}^{2} \overline{A_{0}} + 4 \, A_{0} \alpha_{0} \overline{A_{0}} \overline{\alpha_{0}} - 2 \, A_{1} \overline{A_{0}} \overline{\alpha_{0}} - 2 \, {\left(A_{0} \alpha_{0} - A_{1}\right)} \overline{A_{1}} & \overline{A_{1}} & 0 & 4 & 0 \\
-4 \, \overline{A_{0}}^{2} & A_{2} & 1 & 2 & 0 \\
4 \, \alpha_{0} \overline{A_{0}}^{2} & A_{1} & 1 & 2 & 0 \\
-4 \, {\left(\alpha_{0}^{2} - \alpha_{1}\right)} \overline{A_{0}}^{2} + \frac{1}{6} \, C_{1} \overline{A_{0}} & A_{0} & 1 & 2 & 0 \\
4 \, {\left(\alpha_{0}^{2} - \alpha_{1}\right)} A_{0} \overline{A_{0}} - 4 \, A_{1} \alpha_{0} \overline{A_{0}} + 4 \, A_{2} \overline{A_{0}} & \overline{A_{0}} & 1 & 2 & 0 \\
-\frac{1}{9} \, A_{0} \overline{A_{0}} & B_{1} & 1 & 3 & 0 \\
4 \, \overline{A_{0}}^{2} \overline{\alpha_{0}} - 8 \, \overline{A_{0}} \overline{A_{1}} & A_{2} & 1 & 3 & 0 \\
-\frac{1}{6} \, A_{0} \overline{A_{0}} \overline{\alpha_{0}} + \frac{1}{6} \, A_{0} \overline{A_{1}} & C_{1} & 1 & 3 & 0 \\
4 \, {\left(\alpha_{0}^{2} - \alpha_{1}\right)} A_{0} \overline{A_{0}} - 4 \, A_{1} \alpha_{0} \overline{A_{0}} - \frac{1}{3} \, A_{0} C_{1} + 4 \, A_{2} \overline{A_{0}} & \overline{A_{1}} & 1 & 3 & 0 \\
8 \, A_{1} \alpha_{0} \overline{A_{0}} \overline{\alpha_{0}} + 8 \, A_{0}^{2} \alpha_{2} + 8 \, {\left(2 \, A_{0} \alpha_{0} \overline{A_{0}} - A_{1} \overline{A_{0}}\right)} {\left| A_{1} \right|}^{2} - 4 \, {\left(7 \, \alpha_{0}^{2} \overline{\alpha_{0}} - 2 \, \alpha_{1} \overline{\alpha_{0}} + \alpha_{4}\right)} A_{0} \overline{A_{0}} - 4 \, {\left(\overline{A_{0}} \overline{\alpha_{0}} - \overline{A_{1}}\right)} A_{2} - \frac{1}{18} \, A_{0} B_{1} + \frac{1}{9} \, {\left(4 \, A_{0}^{2} \overline{A_{0}} \overline{\alpha_{0}} - 4 \, A_{0}^{2} \overline{A_{1}} + 3 \, A_{0} \overline{\alpha_{0}}\right)} C_{1} + 4 \, {\left({\left(\alpha_{0}^{2} - \alpha_{1}\right)} A_{0} - A_{1} \alpha_{0}\right)} \overline{A_{1}} & \overline{A_{0}} & 1 & 3 & 0 \\
8 \, {\left| A_{1} \right|}^{2} \overline{A_{0}}^{2} - 8 \, \alpha_{0} \overline{A_{0}}^{2} \overline{\alpha_{0}} + 8 \, \alpha_{0} \overline{A_{0}} \overline{A_{1}} & A_{1} & 1 & 3 & 0 \\
-16 \, \alpha_{0} {\left| A_{1} \right|}^{2} \overline{A_{0}}^{2} - 8 \, A_{0} \alpha_{2} \overline{A_{0}} + 4 \, {\left(7 \, \alpha_{0}^{2} \overline{\alpha_{0}} - 2 \, \alpha_{1} \overline{\alpha_{0}} +
\end{array}
$$

$$
\begin{array}{l}
\alpha_{4}\right)} \overline{A_{0}}^{2} - 8 \, {\left(\alpha_{0}^{2}
- \alpha_{1}\right)} \overline{A_{0}} \overline{A_{1}} - \frac{1}{6}
\, {\left(\overline{A_{0}} \overline{\alpha_{0}} - 2 \,
\overline{A_{1}}\right)} C_{1} + \frac{5}{18} \, B_{1}
\overline{A_{0}} & A_{0} & 1 & 3 & 0 \\
-2 \, A_{0}^{2} & \overline{A_{1}} & 2 & 0 & 0 \\
-2 \, A_{0} \overline{A_{0}} \overline{\alpha_{0}} + 2 \, A_{0}
\overline{A_{1}} & A_{0} & 2 & 0 & 0 \\
2 \, A_{0}^{2} \overline{\alpha_{0}} & \overline{A_{0}} & 2 & 0 & 0
\\
-4 \, A_{0}^{2} & \overline{A_{2}} & 2 & 1 & 0 \\
4 \, A_{0}^{2} \overline{\alpha_{0}} & \overline{A_{1}} & 2 & 1 & 0
\\
4 \, {\left(\overline{\alpha_{0}}^{2} - \overline{\alpha_{1}}
\right)} A_{0} \overline{A_{0}} - 4 \, A_{0} \overline{A_{1}}
\overline{\alpha_{0}} + 4 \, A_{0} \overline{A_{2}} & A_{0} & 2 & 1
& 0 \\
-4 \, {\left(\overline{\alpha_{0}}^{2} - \overline{\alpha_{1}}
\right)} A_{0}^{2} + \frac{1}{6} \, A_{0} \overline{C_{1}} &
\overline{A_{0}} & 2 & 1 & 0 \\
-6 \, \overline{A_{0}}^{2} & A_{3} & 2 & 2 & 0 \\
-6 \, A_{0}^{2} & \overline{A_{3}} & 2 & 2 & 0 \\
-\frac{1}{4} \, A_{0} \overline{A_{0}} \overline{\alpha_{0}} +
\frac{1}{4} \, A_{0} \overline{A_{1}} & \overline{C_{1}} & 2 & 2 & 0
\\
6 \, A_{0}^{2} \overline{\alpha_{0}} & \overline{A_{2}} & 2 & 2 & 0
\\
6 \, \alpha_{0} \overline{A_{0}}^{2} & A_{2} & 2 & 2 & 0 \\
-6 \, {\left(\alpha_{0}^{2} - \alpha_{1}\right)} \overline{A_{0}}
^{2} - \frac{3}{8} \, C_{1} \overline{A_{0}} & A_{1} & 2 & 2 & 0 \\
6 \, {\left(\alpha_{0}^{3} - 2 \, \alpha_{0} \alpha_{1} +
\alpha_{3}\right)} \overline{A_{0}}^{2} + 6 \,
{\left(\overline{\alpha_{0}}^{2} - \overline{\alpha_{1}}\right)}
A_{0} \overline{A_{1}} - 6 \, A_{0} \overline{A_{2}}
\overline{\alpha_{0}} + \frac{1}{6} \, {\left(4 \, A_{0} \alpha_{0}
\overline{A_{0}}^{2} - 4 \, A_{1} \overline{A_{0}}^{2} + 3 \,
\alpha_{0} \overline{A_{0}}\right)} C_{1} - \frac{1}{6} \, {\left(36
\, {\left(\overline{\alpha_{0}}^{3} - 2 \, \overline{\alpha_{0}}
\overline{\alpha_{1}} + \overline{\alpha_{3}}\right)} A_{0} +
\gamma_{1}\right)} \overline{A_{0}} + 6 \, A_{0} \overline{A_{3}} -
\frac{1}{4} \, {\left(\overline{A_{0}} \overline{\alpha_{0}} -
\overline{A_{1}}\right)} \overline{C_{1}} & A_{0} & 2 & 2 & 0 \\
-\frac{1}{4} \, A_{0} \alpha_{0} \overline{A_{0}} + \frac{1}{4} \,
A_{1} \overline{A_{0}} & C_{1} & 2 & 2 & 0 \\
-6 \, {\left(\overline{\alpha_{0}}^{2} - \overline{\alpha_{1}}
\right)} A_{0}^{2} - \frac{3}{8} \, A_{0} \overline{C_{1}} &
\overline{A_{1}} & 2 & 2 & 0 \\
6 \, {\left(\overline{\alpha_{0}}^{3} - 2 \, \overline{\alpha_{0}}
\overline{\alpha_{1}} + \overline{\alpha_{3}}\right)} A_{0}^{2} - 6
\, {\left(\alpha_{0}^{3} - 2 \, \alpha_{0} \alpha_{1} +
\alpha_{3}\right)} A_{0} \overline{A_{0}} + 6 \, {\left(\alpha_{0}
^{2} - \alpha_{1}\right)} A_{1} \overline{A_{0}} - 6 \, A_{2}
\alpha_{0} \overline{A_{0}} - \frac{1}{4} \, {\left(A_{0} \alpha_{0}
- A_{1}\right)} C_{1} - \frac{1}{6} \, A_{0} \gamma_{1} + 6 \, A_{3}
\end{array}
$$

$$
\begin{array}{ccccc}
\overline{A_{0}} + \frac{1}{6} \, {\left(4 \, A_{0}^{2}
\overline{A_{0}} \overline{\alpha_{0}} - 4 \, A_{0}^{2}
\overline{A_{1}} + 3 \, A_{0} \overline{\alpha_{0}}\right)}
\overline{C_{1}} & \overline{A_{0}} & 2 & 2 & 0 \\
-2 \, A_{0} \overline{A_{0}} \overline{\alpha_{0}} + 2 \, A_{0}
\overline{A_{1}} & A_{1} & 3 & 0 & 0 \\
2 \, A_{0}^{2} \alpha_{0} - 4 \, A_{0} A_{1} & \overline{A_{1}} & 3
& 0 & 0 \\
-4 \, A_{0} {\left| A_{1} \right|}^{2} \overline{A_{0}} + 4 \, A_{0}
\alpha_{0} \overline{A_{0}} \overline{\alpha_{0}} - 2 \, A_{1}
\overline{A_{0}} \overline{\alpha_{0}} - 2 \, {\left(A_{0}
\alpha_{0} - A_{1}\right)} \overline{A_{1}} & A_{0} & 3 & 0 & 0 \\
4 \, A_{0}^{2} {\left| A_{1} \right|}^{2} - 4 \, A_{0}^{2}
\alpha_{0} \overline{\alpha_{0}} + 4 \, A_{0} A_{1}
\overline{\alpha_{0}} & \overline{A_{0}} & 3 & 0 & 0 \\
-\frac{1}{9} \, A_{0} \overline{A_{0}} & \overline{B_{1}} & 3 & 1 &
0 \\
-\frac{1}{6} \, A_{0} \alpha_{0} \overline{A_{0}} + \frac{1}{6} \,
A_{1} \overline{A_{0}} & \overline{C_{1}} & 3 & 1 & 0 \\
4 \, {\left(\overline{\alpha_{0}}^{2} - \overline{\alpha_{1}}
\right)} A_{0} \overline{A_{0}} - 4 \, A_{0} \overline{A_{1}}
\overline{\alpha_{0}} + 4 \, A_{0} \overline{A_{2}} - \frac{1}{3} \,
\overline{A_{0}} \overline{C_{1}} & A_{1} & 3 & 1 & 0 \\
8 \, {\left(2 \, A_{0} \overline{A_{0}} \overline{\alpha_{0}} -
A_{0} \overline{A_{1}}\right)} {\left| A_{1} \right|}^{2} - 4 \,
{\left(7 \, \alpha_{0} \overline{\alpha_{0}}^{2} - 2 \, \alpha_{0}
\overline{\alpha_{1}} + \overline{\alpha_{4}}\right)} A_{0}
\overline{A_{0}} + 4 \, {\left(\overline{\alpha_{0}}^{2} -
\overline{\alpha_{1}}\right)} A_{1} \overline{A_{0}} + 8 \,
\overline{A_{0}}^{2} \overline{\alpha_{2}} + 4 \, {\left(2 \, A_{0}
\alpha_{0} \overline{\alpha_{0}} - A_{1} \overline{\alpha_{0}}
\right)} \overline{A_{1}} - 4 \, {\left(A_{0} \alpha_{0} -
A_{1}\right)} \overline{A_{2}} - \frac{1}{18} \, \overline{A_{0}}
\overline{B_{1}} + \frac{1}{9} \, {\left(4 \, A_{0} \alpha_{0}
\overline{A_{0}}^{2} - 4 \, A_{1} \overline{A_{0}}^{2} + 3 \,
\alpha_{0} \overline{A_{0}}\right)} \overline{C_{1}} & A_{0} & 3 & 1
& 0 \\
4 \, A_{0}^{2} \alpha_{0} - 8 \, A_{0} A_{1} & \overline{A_{2}} & 3
& 1 & 0 \\
8 \, A_{0}^{2} {\left| A_{1} \right|}^{2} - 8 \, A_{0}^{2}
\alpha_{0} \overline{\alpha_{0}} + 8 \, A_{0} A_{1}
\overline{\alpha_{0}} & \overline{A_{1}} & 3 & 1 & 0 \\
-16 \, A_{0}^{2} {\left| A_{1} \right|}^{2} \overline{\alpha_{0}} +
4 \, {\left(7 \, \alpha_{0} \overline{\alpha_{0}}^{2} - 2 \,
\alpha_{0} \overline{\alpha_{1}} + \overline{\alpha_{4}}\right)}
A_{0}^{2} - 8 \, {\left(\overline{\alpha_{0}}^{2} -
\overline{\alpha_{1}}\right)} A_{0} A_{1} - 8 \, A_{0}
\overline{A_{0}} \overline{\alpha_{2}} + \frac{5}{18} \, A_{0}
\overline{B_{1}} - \frac{1}{6} \, {\left(A_{0} \alpha_{0} - 2 \,
A_{1}\right)} \overline{C_{1}} & \overline{A_{0}} & 3 & 1 & 0 \\
-\frac{11}{36} \, A_{0}^{2} & B_{1} & 4 & 0 & 0 \\
-2 \, A_{0} \overline{A_{0}} \overline{\alpha_{0}} + 2 \, A_{0}
\overline{A_{1}} & A_{2} & 4 & 0 & 0 \\
-2 \, {\left(\alpha_{0}^{2} - \alpha_{1}\right)} A_{0}^{2} + 4 \,
\end{array}
$$

```
A_{0} A_{1} \alpha_{0} - 2 \, A_{1}^{2} - 4 \, A_{0} A_{2} &
\overline{A_{1}} & 4 & 0 & 0 \\
4 \, A_{1} \alpha_{0} \overline{A_{0}} \overline{\alpha_{0}} + 4 \,
{\left(2 \, A_{0} \alpha_{0} \overline{A_{0}} - A_{1}
\overline{A_{0}}\right)} {\left| A_{1} \right|}^{2} - 2 \, {\left(19
\, \alpha_{0}^{2} \overline{\alpha_{0}} - 2 \, \alpha_{1}
\overline{\alpha_{0}} + \alpha_{4}\right)} A_{0} \overline{A_{0}} -
2 \, {\left(\overline{A_{0}} \overline{\alpha_{0}} -
\overline{A_{1}}\right)} A_{2} + \frac{7}{36} \, A_{0} B_{1} +
\frac{5}{9} \, {\left(A_{0}^{2} \overline{A_{0}}
\overline{\alpha_{0}} - A_{0}^{2} \overline{A_{1}}\right)} C_{1} + 2
\, {\left({\left(\alpha_{0}^{2} - \alpha_{1}\right)} A_{0} - A_{1}
\alpha_{0}\right)} \overline{A_{1}} & A_{0} & 4 & 0 & 0 \\
-8 \, A_{0} A_{1} \alpha_{0} \overline{\alpha_{0}} + 2 \, {\left(19
\, \alpha_{0}^{2} \overline{\alpha_{0}} - 2 \, \alpha_{1}
\overline{\alpha_{0}} + \alpha_{4}\right)} A_{0}^{2} - 8 \,
{\left(A_{0}^{2} \alpha_{0} - A_{0} A_{1}\right)} {\left| A_{1}
\right|}^{2} + 2 \, A_{1}^{2} \overline{\alpha_{0}} + 4 \, A_{0}
A_{2} \overline{\alpha_{0}} & \overline{A_{0}} & 4 & 0 & 0 \\
-4 \, A_{0} {\left| A_{1} \right|}^{2} \overline{A_{0}} + 4 \, A_{0}
\alpha_{0} \overline{A_{0}} \overline{\alpha_{0}} - 2 \, A_{1}
\overline{A_{0}} \overline{\alpha_{0}} - 2 \, {\left(A_{0}
\alpha_{0} - A_{1}\right)} \overline{A_{1}} & A_{1} & 4 & 0 & 0 \\
\frac{1}{6} \, A_{0}^{2} & \overline{C_{1}} & 5 & -2 & 0 \\
-\frac{1}{6} \, A_{0}^{2} & \gamma_{1} & 5 & -1 & 0 \\
-16 \, A_{0} \alpha_{0}^{3} \overline{A_{0}} + \frac{2}{3} \,
{\left(A_{0}^{2} \overline{A_{0}} \overline{\alpha_{0}} - A_{0}^{2}
\overline{A_{1}}\right)} \overline{C_{1}} & A_{0} & 5 & -1 & 0 \\
16 \, A_{0}^{2} \alpha_{0}^{3} & \overline{A_{0}} & 5 & -1 & 0 \\
-\frac{1}{24} \, A_{0} \overline{C_{1}} & A_{1} & 6 & -2 & 0 \\
\frac{5}{72} \, A_{0}^{2} & \overline{B_{1}} & 6 & -2 & 0 \\
2 \, A_{0} \overline{A_{0}} \overline{\alpha_{2}} + \frac{11}{72} \,
A_{0} \overline{B_{1}} - \frac{1}{72} \, {\left(16 \, A_{0}^{2}
\alpha_{0} \overline{A_{0}} - 16 \, A_{0} A_{1} \overline{A_{0}} - 3
\, A_{0} \alpha_{0}\right)} \overline{C_{1}} & A_{0} & 6 & -2 & 0 \\
-2 \, A_{0}^{2} \overline{\alpha_{2}} + \frac{1}{9} \, {\left(A_{0}
^{3} \alpha_{0} - A_{0}^{2} A_{1}\right)} \overline{C_{1}} &
\overline{A_{0}} & 6 & -2 & 0 \\
-\frac{1}{24} \, A_{0}^{2} \alpha_{0} + \frac{1}{3} \, A_{0} A_{1} &
\overline{C_{1}} & 6 & -2 & 0
\end{array}\right)
}}}

# From now on we assume that alpha_0=0 and we compute again each
tensor, we indicate each change by #change alpha_0=0

n=3

g=matrix([[1,n-1,n-1,0],[2*abs(a1)^2,n,n,0],[alpha1,n+1,n-1,0],
[alpha1b,n-1,n+1,0]])

latex(g)

H2=real_part2(matrix([[1,c1,2-n,0,0],[1,b1,-1,1,0],[1,c2,0,0,0],
```

```
[1,gamma1,0,0,1]]))

dzphi2=sum_matrix(matrix([[1,a0,n-1,0,0],[1,a1,n,0,0],
[1,a2,n+1,0,0],
[1,a3,n+2,0,0]]),mult_scalar(1/2,matrix_sort(matrix_full_simplify(in
tzb(throw(prod(g,H2),n+2))))))

latex(dzphi2)

phi=sum_matrix(matrix([[1/n,a0b,0,n,0],[1/(n+1),a1b,0,n+1,0],[1/
(n+2),a2b,0,n+2,0],[1/(n+3),a3b,0,n+3,0]]),intz(dzphi2))

latex(phi)

g2=sum_matrix(matrix([[1,n-1,n-1,0],
[2*abs(a1)^2,n,n,0]]),real_part2(matrix([[2*alpha1,n+1,n-1,0],
[2*alpha2,1,2*n,0],[2*alpha3,5,2,0],[2*alpha4,4,3,0]])))

latex(g2)

g2inv=power_divide(taylor_inverse(power_divide(g2,n-1,n-1),3,5),n-1,
n-1)

h02=Weingarten(dzphi2,g2,g2inv,3,n+2)

latex(h02)

H5m1=H2

H5=H2

m1=throw(prod(scal(H2,H2),throw(dzphi2,n)),4-n)

h02m2=throw(h02,n+1)

H5m2=throw(H5,4-n)

g2invm2=throw(g2inv,4-2*n)

dzphi2m2=throw(dzphi2,n+1)

m2=throw(mult_scalar(2,prod(scal(H5m2,h02m2),prod(g2invm2,bar(dzphi2
m2)))),4-n)

alpha=factor_simplify(sum_matrix(diffz(H5),sum_matrix(m1,m2)))

latex(alpha)


#####


Q=intQ(dzphi2,g2inv,H2,h02,5-n)
```

```
Htemp=matrix_sort(real_part2(Q))

latex(Htemp)

#####

g2=sum_matrix(matrix([[1,n-1,n-1,0],
[2*abs(a1)^2,n,n,0]]),real_part2(matrix([[2*alpha1,n+1,n-1,0],
[2*alpha2,1,2*n,0],[2*alpha3,5,2,0],[2*alpha4,4,3,0]])))

H3=real_part2(matrix([[1,c1,2-n,0,0],[1,c2,0,0,0],[1,c3,1,0,0],
[1,b1,-1,1,0],[1,b2,-1,2,0],[1,e1,-2,3,0],[1,gamma1,0,0,1],
[1,gamma2,1,0,1]]))

latex(H3)

dzphi3=sum_matrix(matrix([[1,a0,n-1,0,0],[1,a1,n,0,0],
[1,a2,n+1,0,0],[1,a3,n+2,0,0],
[1,a4,n+3,0,0]]),mult_scalar(1/2,matrix_sort(matrix_full_simplify(in
tzb(throw(prod(g2,H3),n+3))))))

latex(dzphi3)

conf=matrix_full_simplify(throw(scal(dzphi3,dzphi3),2*n+3))

latex(conf)

g3temp=mult_scalar(2,matrix_full_simplify(throw(scal(dzphi3,bar(dzph
i3)),2*n+3)))

latex(g3temp)

g3=sum_matrix(matrix([[1,n-1,n-1,0],[2*abs(a1)^2,n,n,0],
[beta,4,4,0]]),real_part2(matrix([[2*alpha1,n+1,n-1,0],
[2*alpha2,1,2*n,0],[2*alpha3,5,2,0],[2*alpha4,4,3,0],
[2*alpha5,7,1,0],[2*alpha6,6,2,0],[2*alpha7,3,5,0],
[2*zeta0,6,2,1]])))

latex(g3)

g3inv=power_divide(taylor_inverse(power_divide(g3,n-1,n-1),3,5),n-1,
n-1)

var('gamma2',domain='complex')

h03=Weingarten_simpler(dzphi3,g3,g3inv,4,n+3)

latex(matrix_sort(h03))

Q2=intQ(dzphi3,g3inv,H3,h03,6-n)

Htemp2=matrix_sort(real_part2(Q2))

latex(Htemp2)
```

```
H4=H3=real_part2(matrix([[1,c1,2-n,0,0],[1,c2,3-n,0,0],[1,c3,4-
n,0,0],[1,c4,5-n,0,0],[1,b1,-1,1,0],[1,b2,-1,2,0],[1,b3,-1,3,0],
[1,b4,1,1,0],[1,e1,-2,3,0],[1,e2,-2,4,0],[1,gamma1,0,0,1],
[1,gamma2,1,0,1],[1,gamma3,2,0,1],[1,gamma4,-1,3,0],
[1,gamma5,1,1,1]]))

latex(H4)

dzphi4=sum_matrix(matrix([[1,a0,n-1,0,0],[1,a1,n,0,0],
[1,a2,n+1,0,0],[1,a3,n+2,0,0],[1,a4,n+3,0,0],
[1,a5,n+4,0,0]]),mult_scalar(1/2,matrix_sort(matrix_full_simplify(in
tzb(throw(prod(g3,H4),n+4))))))

latex(dzphi4)

conf=matrix_sort(matrix_full_simplify(throw(scal(dzphi4,dzphi4),2*n+
4)))

latex(conf)

g4temp=mult_scalar(2,matrix_sort(matrix_full_simplify(throw(scal(dzp
hi4,bar(dzphi4)),2*n+4))))

latex(g4temp)

g4=sum_matrix(matrix([[1,n-1,n-1,0],[2*abs(a1)^2,n,n,0],
[beta,4,4,0]]),real_part2(matrix([[2*alpha1,n+1,n-1,0],
[2*alpha2,1,2*n,0],[2*alpha3,5,2,0],[2*alpha4,4,3,0],
[2*alpha5,7,1,0],[2*alpha6,6,2,0],[2*alpha7,3,5,0],[2*alpha8,8,1,0],
[2*alpha9,7,2,0],[2*alpha10,6,3,0],[2*alpha11,5,4,0],
[2*zeta0,6,2,1],[2*zeta1,7,2,1],[2*zeta2,6,3,1],[2*zeta3,5,4,1]])))

latex(g4)

g4inv=power_divide(taylor_inverse(power_divide(g3,n-1,n-1),3,6),n-1,
n-1)

h04=Weingarten_simpler(dzphi4,g4,g4inv,5,n+4)

latex(h04)

quartic_end=sing_quartic(throw(g4inv,6-2*n),h04,3)

latex(quartic_end)

### Gauss curvature

e2uinv=power_divide(throw(g4inv,5-2*n),1-n,1-n) ## this tensor is
not derived, so we need only a development up to order 3

latex(e2uinv)

e2u=power_divide(throw(g4,2*n-2+5),n-1,n-1) # we take two
```

```
derivatives, so we need a development up to order 5

latex(e2u)

h04K=throw(h04,n-1+3) ## this tensor is not derived, so we need only
a development up to order 3

g4invK=throw(g4inv,2-2*n+3) ## this tensor is not derived, so we
need only a development up to order 3

K=Gauss_Weingarten(e2u,e2uinv,g4invK,h04K,3)

latex(K)

## final development of the quartic form, modulo the easy computed
\frac{5}{4}|\H|^2\h_0\totimes \h_0+\s{\H}{\h_0}^2

Q_end=matrix_full_simplify(sum_matrix(quartic_end,K))

latex(Q_end)
```