In [608]:
```
1+1+1+4+1
```

Out[608]: 8

In [609]:
```
### program for standing waves based on theory of Sobey (2009)
### for publication in ICE journal
### code written in SAGE (which is an open source and free software)
### the output must be suppressed wherever possible to save time
```

In [610]:
```
reset()
```

In [611]:
```
#populating variables with itself
```

In [612]:
```
x, y, z, t=var('x y z t')
ii=var('ii')
assume(ii,'integer')
i, j, m=var('i j m')
q=var('q')
eps=var('eps')

A =[[[0 for i in range(1,7)] for j in range(7)] for m in range(7)]
B =[[[0 for i in range(1,7)] for j in range(7)] for m in range(7)]
C=[0 for i in range (1,7)]
D=[0 for i in range(1,7)]
for i in range(1,6):
    C[i]=var('cc_'+str(i))
    D[i]=var('dd_'+str(i))
    for j in range(6):
        for m in range(6):
            A[i][j][m]=var('aa_'+str(i)+'_'+str(j)+'_'+str(m))
            B[i][j][m]=var('bb_'+str(i)+'_'+str(j)+'_'+str(m))
#the variables are of the form A[i][j][m] and B[i][j][m] which are populated with aa_i_j
#also C[i] and D[i] are populated with cc_i and dd_i
```

In [ ]:
```

```

In [613]:
```
#test values using random numbers
#eps=0.09655585; x=0.732980018; t=0.410875873;q=0.855298308;
#B[1][0][0]=0.0491999314581644; B[1][0][1]=0.0447561933639101; B[1][0][2]=0.055645533028
#D[1] = 0.173877213228345; D[2] = 0.119787287086215;
#D[3] = 0.157160772869084; D[4] = 0.117340645532323;
```

In [614]:
```
print(C[1],C[2],C[3],C[4],C[5])
```

```
(cc_1, cc_2, cc_3, cc_4, cc_5)
```

In [615]:
```
def fw(N):
    temp=0
    for i in range(1,N+1):
        temp=temp+C[i]*eps^(i-1)
    return temp

def fB(N):
    temp=0
    for i in range(1,N+1):
        temp=temp+D[i]*eps^i
    return temp

def fq(j,q):
    return tanh(j*arctanh(q)).trig_expand()

print("fq values")
for j in range(6):
    print('j='+str(j),fq(j,q)),
```

```
fq values
('j=0', 0) ('j=1', q) ('j=2', 2*q/(q^2 + 1)) ('j=3', (q^3 + 3*q)/(3*q^2 + 1)) ('j=4',
4*(q^3 + q)/(q^4 + 6*q^2 + 1)) ('j=5', (q^5 + 10*q^3 + 5*q)/(5*q^4 + 10*q^2 + 1))
```

In [616]:
```
def f_eta(x,t,N):
    temp=0
    for i in range(1,N+1):#counter one more than necessary
        for j in range(i+1):
            for m in range(i+1):
                temp=temp+eps^i*B[i][j][m]*cos(j*x)*cos(m*t)
    return temp

def f_eta_t(x,t,N):
    temp=0
    for i in range(1,N+1):
        for j in range(i+1):
            for m in range(i+1):
                temp=temp-m*fw(N-i+1)*eps^i*B[i][j][m]*cos(j*x)*sin(m*t)
    return temp

def f_eta_x(x,t,N):
    temp=0
    for i in range(1,N+1):
        for j in range(i+1):
            for m in range(i+1):
                temp=temp-j*eps^i*B[i][j][m]*sin(j*x)*cos(m*t)
    return temp
```

In [617]:
```
def fcosh(x,t,N,i,j):
    temp1=1
    temp2=0
    for nn in range(1,N):
        temp00=f_eta(x,t,floor((N-i)/nn+1))
        if mod(nn,2)==0:
            temp1=temp1+(1+(-1)^nn)/2/factorial(nn)*(j*temp00)^nn
        if mod(nn,2)==1:
            temp2=temp2+(1-(-1)^nn)/2/factorial(nn)*(j*temp00)^nn
    return temp1+temp2*fq(j,q)

def fcoshP(x,t,N,i,j):
    temp1=1
    temp2=0
    for nn in range(1,N):
        temp00=f_eta(x,t,floor((N-i)/nn+1))
        if mod(nn,2)==0:
            temp1=temp1+(1+(-1)^nn)/2/factorial(nn)*(j*temp00)^nn
        if mod(nn,2)==1:
            temp2=temp2+(1-(-1)^nn)/2/factorial(nn)*(j*temp00)^nn
    return temp2+temp1*fq(j,q)
```

In [618]:
```
def f_phi(x,t,N):
    temp=0
    for i in range(1,N+1):
        for j in range(i+1):
            for m in range(i+1):
                temp=temp+eps^i*A[i][j][m]*cos(j*x)*sin(m*t)*fcosh(x,t,N,i,j)
    return temp

def f_phi_t(x,t,N):
    temp=0
    for i in range(1,N+1):
        for j in range(i+1):
            for m in range(i+1):
                temp=temp+eps^i*m*fw(N-i+1)*A[i][j][m]*cos(j*x)*cos(m*t)*fcosh(x,t,N,i,j
    return temp

def f_u(x,t,N):
    temp=0
    for i in range(1,N+1):
        for j in range(i+1):
            for m in range(i+1):
                temp=temp-eps^i*j*A[i][j][m]*sin(j*x)*sin(m*t)*fcosh(x,t,N,i,j)
    return temp

def f_w(x,t,N):
    temp=0
    for i in range(1,N+1):
        for j in range(i+1):
            for m in range(i+1):
                temp=temp+j*eps^i*A[i][j][m]*cos(j*x)*sin(m*t)*fcoshP(x,t,N,i,j)
    return temp
```

In [619]:
```
#lateral boundary conditions yield nothing:
f_phi(x,t,5)-f_phi(x,t+2*pi,5)
```

Out[619]: 0

In [620]:
```
f_phi(x,t,5)-f_phi(x+2*pi,t,5)
```

Out[620]: 0

In [621]:
```
f_eta(x,t,5)-f_eta(x+2*pi,t,5)
```

Out[621]:
```
0
```

In [622]:
```
f_eta(x,t,5)-f_eta(x,t+2*pi,5)
```

Out[622]:
```
0
```

In [623]:
```
#continuity equation
integral(f_eta(x,t,5),x,0,2*pi)
```

Out[623]:
```
2*pi*bb_5_0_5*eps^5*cos(5*t) + 2*pi*bb_5_0_0*eps^5 + 2*pi*bb_4_0_0*eps^4 + 2*pi*bb_3_0
_0*eps^3 + 2*pi*bb_2_0_0*eps^2 + 2*pi*bb_1_0_0*eps + 2*(pi*bb_5_0_4*eps^5 + pi*bb_4_0_
4*eps^4)*cos(4*t) + 2*(pi*bb_5_0_3*eps^5 + pi*bb_4_0_3*eps^4 + pi*bb_3_0_3*eps^3)*cos
(3*t) + 2*(pi*bb_5_0_2*eps^5 + pi*bb_4_0_2*eps^4 + pi*bb_3_0_2*eps^3 + pi*bb_2_0_2*eps
^2)*cos(2*t) + 2*(pi*bb_5_0_1*eps^5 + pi*bb_4_0_1*eps^4 + pi*bb_3_0_1*eps^3 + pi*bb_2_
0_1*eps^2 + pi*bb_1_0_1*eps)*cos(t)
```

In [624]:
```
for i in range(1,5+1):
    for j in range(5+1):
        for m in range(5+1):
            if j==0:
                B[i][j][m]=0
                vars()['bb_'+str(i)+'_'+str(j)+'_'+str(m)]=0
            #additional constraints
            if m==0:
                A[i][j][m]=0
                vars()['aa_'+str(i)+'_'+str(j)+'_'+str(m)]=0
            if mod(j+m,2)==1 or mod(i+j,2)==1 or mod(i+m,2)==1:
                B[i][j][m]=0
                vars()['bb_'+str(i)+'_'+str(j)+'_'+str(m)]=0
                A[i][j][m]=0
                vars()['aa_'+str(i)+'_'+str(j)+'_'+str(m)]=0
```

In [625]:
```
##########################################################################################
```

In [626]:
```
##########################################################################################
```

In [627]:
```
################################          order 1                    ####################
```

In [628]:
```
##########################################################################################
```

In [629]:
```
##########################################################################################
```

In [630]:
```
##########################################################################################
```

In [ ]:
```

```

In [631]:
```
#order 1 analysis#
#f_eta(x,t,Neta,Nw)
#wave height constraint 1, order 1
```

In [632]:
```
eq_a00=expand((f_eta(0,0,1)-f_eta(pi,0,1)-2*eps)/(2*eps))
print(eq_a00)
```

```
bb_1_1_1 - 1
```

In [633]:
```
#wave height constraint 2, order1
eq_a01=expand((f_eta(0,0,1)-f_eta(0,pi,1)-2*eps)/(2*eps))
print(eq_a01)
```

```
bb_1_1_1 - 1
```

In [634]:
```
##########################################################################################
#fw(x,t,Nphi,Nt,Neta,Nw)
#Kinematic BC, order 1
```

In [635]:
```
eq_a1=f_eta_t(x,t,1)-f_w(x,t,1)
print(eq_a1)
```

```
-bb_1_1_1*cc_1*eps*cos(x)*sin(t) - aa_1_1_1*eps*q*cos(x)*sin(t)
```

In [636]:
```
#for using .coefficient() function, perform .expand() first
eq_a_temp1=eq_a1.expand().coefficient(eps,1)
eq_a2=expand(eq_a_temp1/(sin(t)*cos(x)))
print(eq_a2)
```

```
-bb_1_1_1*cc_1 - aa_1_1_1*q
```

In [637]:
```
##########################################################################################
#dynamic BC, order1
```

In [638]:
```
eq_ct_1=((f_phi_t(x,t,1)+f_eta(x,t,1)-fB(1))).expand().coefficient(eps,1)
print(eq_ct_1)
```

```
aa_1_1_1*cc_1*cos(t)*cos(x) + bb_1_1_1*cos(t)*cos(x) - dd_1
```

In [639]:
```
#obtain coefficients of double fourier series of the form a_kl*sin(kx)*sin(lt) using int
const_=4
eq_ct_11=[0 for i in range(6)]
for n1 in range(1+1):
    for n2 in range (1+1):
        eq_ct_11[n1*2+n2]=expand(integral(integral(eq_ct_1*cos(n1*x)*cos(n2*t)/pi^2,x,0,
        print(eq_ct_11[n1*2+n2])
```

```
-4*dd_1
0
0
aa_1_1_1*cc_1 + bb_1_1_1
```

In [640]:
```
#######################                combine BCs-01         ##########################
```

In [641]:
```
#the wave height constraint, kinematic and dynamic boundary conditions are written in on
eq_order1=[0 for i in range(4)]
eq_order1[0]=eq_a00
eq_order1[1]=eq_a2
eq_order1[2]=eq_ct_11[0]
eq_order1[3]=eq_ct_11[3]

print(eq_order1)
```

```
[bb_1_1_1 - 1, -bb_1_1_1*cc_1 - aa_1_1_1*q, -4*dd_1, aa_1_1_1*cc_1 + bb_1_1_1]
```

In [642]:
```
#######################                list of vars-01         ##########################
```

In [643]:
```
#flattening the equations to remove the list property to be able to extract variable nam
eps1=var('eps1')
temp=0
for i in range(len(eq_order1)):
    temp=temp+eq_order1[i]*eps1^i

#list of variable names to be used in solve
list_var=temp.variables()

#the variable list should not have eps1 & q, therefore order reduction will be done
list_var2=[0 for i in range(len(list_var)-2)]

#copy variable list, leave out eps1 & q
ii=var('ii')
ii=0
for i in range(len(list_var)):
    if list_var[i]!=eps1 and list_var[i]!=q:
        list_var2[ii]=list_var[i]
        ii=ii+1

print(list_var2)
```

```
[aa_1_1_1, bb_1_1_1, cc_1, dd_1]
```

In [644]:
```
len(list_var2),len(eq_order1)
```

Out[644]:
```
(4, 4)
```

In [645]:
```
#solve the equation, two solutions [1] and [0] are obtained, solution [0] is meaningful,
sol=solve(eq_order1,list_var2)[0][:]

print(sol)
```

```
[aa_1_1_1 == -1/sqrt(q), bb_1_1_1 == 1, cc_1 == sqrt(q), dd_1 == 0]
```

In [646]:
```
#
```

In [647]:
```
#from list of solve results, compare left hand sides of expressions to variable names aa
#if there was a match, assign the right hand side of solve results to A, B, C, D
#therefore aa, bb, cc and dd are intact
for ii in range(len(sol)):
    for i in range(1,6):
        if var('cc_'+str(i))==sol[ii].lhs():
            C[i]=sol[ii].rhs()
        if var('dd_'+str(i))==sol[ii].lhs():
            D[i]=sol[ii].rhs()
        for j in range(6):
            for m in range(6):
                if var('aa_'+str(i)+'_'+str(j)+'_'+str(m))==sol[ii].lhs():
                    A[i][j][m]=sol[ii].rhs()
                if var('bb_'+str(i)+'_'+str(j)+'_'+str(m))==sol[ii].lhs():
                    B[i][j][m]=sol[ii].rhs()
```

In [648]:
```
print("A[1,1,1]",A[1][1][1])
print("B[1,1,1]",B[1][1][1])
print("C[1]",C[1],"D[1]",D[1])
```

```
('A[1,1,1]', -1/sqrt(q))
('B[1,1,1]', 1)
('C[1]', sqrt(q), 'D[1]', 0)
```

In [649]:
```
##############################        verify-01        ##############################
```

In [650]:
```
#eq_a1
(f_eta_t(x,t,1)-f_w(x,t,1)).expand().coefficient(eps,1)
```

Out[650]: 0

In [651]:
```
#eq_ct_1=
(f_phi_t(x,t,1,)+f_eta(x,t,1)-fB(1)).expand().coefficient(eps,1)
```

Out[651]: 0

In [652]:
```
##########################################################################################
```

In [653]:
```
##########################################################################################
```

In [654]:
```
################################          order 2                    ####################
```

In [655]:
```
##########################################################################################
```

In [656]:
```
##########################################################################################
```

In [657]:
```
#
```

In [658]:
```
########################         wave height constraint order 2          ###############
```

In [659]:
```
eq_aa20=expand((f_eta(0,0,2)-f_eta(pi,0,2)-2*eps)/(2*eps^2))
print(eq_aa20)
```

```
0
```

In [660]:
```
eq_aa21=expand((f_eta(0,0,2)-f_eta(0,pi,2)-2*eps)/(2*eps^2))
print(eq_aa21)
```

```
0
```

In [661]:
```
########################                kinematic BC order 2          #################
```

In [662]:
```
eq_aa1=f_eta_t(x,t,2)+f_u(x,t,2)*f_eta_x(x,t,2)-f_w(x,t,2)

eq_aa2=eq_aa1.expand().coefficients(eps)

for i in range(len(eq_aa2)):
    print(eq_aa2[i][1])
```

```
2
3
4
5
```

In [663]:
```
eq_aa10=eq_aa2[0][0]
print(eq_aa10)
```

```
-2*bb_2_2_2*sqrt(q)*cos(2*x)*sin(2*t) + cos(t)*cos(x)^2*sin(t)/sqrt(q) - cos(t)*sin(t)
*sin(x)^2/sqrt(q) - 4*aa_2_2_2*q*cos(2*x)*sin(2*t)/(q^2 + 1) - cc_2*cos(x)*sin(t)
```

In [664]:
```
#obtain coefficients of double fourier series of the form a_kl*sin(kx)*sin(lt) using int
counter_=0
temp=0
eq_ct_11=[0 for i in range(36)]
for n1 in range(2+1):
    for n2 in range (2+1):
        if n1!=0 and n2!=0:
            temp=expand(integral(integral(eq_aa10*sin(n1*x)*sin(n2*t)/pi^2,x,0,2*pi),t,0
        if temp!=0:
            eq_ct_11[counter_]=temp
            counter_=counter_+1
        if n1!=0:
            temp=expand(integral(integral(eq_aa10*sin(n1*x)*cos(n2*t)/pi^2,x,0,2*pi),t,0
        if temp!=0:
            eq_ct_11[counter_]=temp
            counter_=counter_+1
        if n2!=0:
            temp=expand(integral(integral(eq_aa10*cos(n1*x)*sin(n2*t)/pi^2,x,0,2*pi),t,0
        if temp!=0:
            eq_ct_11[counter_]=temp
            counter_=counter_+1
        temp=expand(integral(integral(eq_aa10*cos(n1*x)*cos(n2*t)/pi^2,x,0,2*pi),t,0,2*p
        if temp!=0:
            eq_ct_11[counter_]=temp
            counter_=counter_+1
eq_ct_12=[0 for i in range(counter_)]
for i in range(counter_):
    eq_ct_12[i]=eq_ct_11[i]
```

In [665]:
```
for i in range(len(eq_ct_12)):
    print(eq_ct_12[i])
```

```
-cc_2
-2*bb_2_2_2*q^(5/2)/(q^2 + 1) - 4*aa_2_2_2*q/(q^2 + 1) - 2*bb_2_2_2*sqrt(q)/(q^2 + 1)
+ 1/2*q^(3/2)/(q^2 + 1) + 1/2/((q^2 + 1)*sqrt(q))
```

In [666]:
```
######################                dynamic BC order 2          ####################
```

In [667]:
```
eq_aa30=f_phi_t(x,t,2)+1/2*(f_u(x,t,1)^2+f_w(x,t,1)^2)+f_eta(x,t,2)-fB(2)

eq_aa31=eq_aa30.expand().coefficients(eps)

for i in range(len(eq_aa2)):
    print(eq_aa2[i][1])
```

```
2
3
4
5
```

In [668]:
```
eq_aa32=eq_aa31[0][0]
print(eq_aa32)
```

```
-q*cos(t)^2*cos(x)^2 + 1/2*q*cos(x)^2*sin(t)^2 + 2*aa_2_2_2*sqrt(q)*cos(2*t)*cos(2*x)
+ bb_2_2_2*cos(2*t)*cos(2*x) + 1/2*sin(t)^2*sin(x)^2/q + 2*aa_2_0_2*sqrt(q)*cos(2*t) -
cc_2*cos(t)*cos(x)/sqrt(q) + bb_2_2_0*cos(2*x) - dd_2
```

In [669]:
```
#obtain coefficients of double fourier series of the form a_kl*sin(kx)*sin(lt) using int
counter_=0
temp=0
eq_aa33=[0 for i in range(36)]
for n1 in range(2+1):
    for n2 in range (2+1):
        if n1!=0 and n2!=0:
            temp=expand(integral(integral(eq_aa32*sin(n1*x)*sin(n2*t)/pi^2,x,0,2*pi),t,0
        if temp!=0:
            eq_aa33[counter_]=temp
            counter_=counter_+1
        if n1!=0:
            temp=expand(integral(integral(eq_aa32*sin(n1*x)*cos(n2*t)/pi^2,x,0,2*pi),t,0
        if temp!=0:
            eq_aa33[counter_]=temp
            counter_=counter_+1
        if n2!=0:
            temp=expand(integral(integral(eq_aa32*cos(n1*x)*sin(n2*t)/pi^2,x,0,2*pi),t,0
        if temp!=0:
            eq_aa33[counter_]=temp
            counter_=counter_+1
        temp=expand(integral(integral(eq_aa32*cos(n1*x)*cos(n2*t)/pi^2,x,0,2*pi),t,0,2*p
        if temp!=0:
            eq_aa33[counter_]=temp
            counter_=counter_+1
eq_aa_34=[0 for i in range(counter_)]
for i in range(counter_):
    eq_aa_34[i]=eq_aa33[i]
```

In [670]:
```
eq_aa_34
```

Out[670]:
```
[-4*dd_2 - 1/2*q + 1/2/q,
 -4*dd_2 - 1/2*q + 1/2/q,
 -4*dd_2 - 1/2*q + 1/2/q,
 4*aa_2_0_2*sqrt(q) - 3/4*q - 1/4/q,
 4*aa_2_0_2*sqrt(q) - 3/4*q - 1/4/q,
 -cc_2/sqrt(q),
 2*bb_2_2_0 - 1/4*q - 1/4/q,
 2*aa_2_2_2*sqrt(q) + bb_2_2_2 - 3/8*q + 1/8/q]
```

In [671]:
```
########################                combine BCs-O2            ########################
```

In [672]:
```
counter_=0
eq_order=[0 for i in range(2+len(eq_ct_12)+len(eq_aa_34))]
for i in range(len(eq_ct_12)):
    eq_order[i]=eq_ct_12[i]
for i in range(len(eq_aa_34)):
    eq_order[i+len(eq_ct_12)]=eq_aa_34[i]
```

In [673]:
```
eq_order
```

Out[673]:
```
[-cc_2,
 -2*bb_2_2_2*q^(5/2)/(q^2 + 1) - 4*aa_2_2_2*q/(q^2 + 1) - 2*bb_2_2_2*sqrt(q)/(q^2 + 1)
+ 1/2*q^(3/2)/(q^2 + 1) + 1/2/((q^2 + 1)*sqrt(q)),
 -4*dd_2 - 1/2*q + 1/2/q,
 -4*dd_2 - 1/2*q + 1/2/q,
 -4*dd_2 - 1/2*q + 1/2/q,
 4*aa_2_0_2*sqrt(q) - 3/4*q - 1/4/q,
 4*aa_2_0_2*sqrt(q) - 3/4*q - 1/4/q,
 -cc_2/sqrt(q),
 2*bb_2_2_0 - 1/4*q - 1/4/q,
 2*aa_2_2_2*sqrt(q) + bb_2_2_2 - 3/8*q + 1/8/q,
 0,
 0]
```

In [674]:
```
########################              list of vars-02        ##########################
```

In [675]:
```
#flattening the equations to remove the list property to be able to extract variable nam
eps1=var('eps1')
temp=0
for i in range(len(eq_order)):
    temp=temp+eq_order[i]*eps1^i
```

In [676]:
```
list_var=temp.variables()
```

In [677]:
```
#list_var
```

In [678]:
```
list_var2=[0 for i in range(len(list_var)-2)]
```

In [679]:
```
#copy variable list, leave out eps1 & q
ii=var('ii')
ii=0
for i in range(len(list_var)):
    if list_var[i]!=eps1 and list_var[i]!=q:
        list_var2[ii]=list_var[i]
        ii=ii+1
```

In [680]:
```
list_var2
```

Out[680]:
```
[aa_2_0_2, aa_2_2_2, bb_2_2_0, bb_2_2_2, cc_2, dd_2]
```

In [681]:
```
len(eq_order),len(list_var2)
```

Out[681]:
```
(12, 6)
```

In [682]:
```
sol=solve(eq_order,list_var2)[0][:]
```

In [683]:
```
sol
```

Out[683]:
```
[aa_2_0_2 == 1/16*(3*q^2 + 1)/q^(3/2),
 aa_2_2_2 == 3/16*(q^4 - 1)/q^(7/2),
 bb_2_2_0 == 1/8*(q^2 + 1)/q,
 bb_2_2_2 == -1/8*(q^2 - 3)/q^3,
 cc_2 == 0,
 dd_2 == -1/8*(q^2 - 1)/q]
```

In [684]:
```
sol[1].lhs()
```

Out[684]:
```
aa_2_2_2
```

In [685]:
```
######################          var substitution-O2         ########################
```

In [686]:
```
#from list of solve results, compare left hand sides of expressions to variable names aa
#if there was a match, assign the right hand side of solve results to A, B, C, D
#therefore aa, bb, cc and dd are intact
for ii in range(len(sol)):
    for i in range(1,6):
        if var('cc_'+str(i))==sol[ii].lhs():
            C[i]=sol[ii].rhs()
        if var('dd_'+str(i))==sol[ii].lhs():
            D[i]=sol[ii].rhs()
        for j in range(6):
            for m in range(6):
                if var('aa_'+str(i)+'_'+str(j)+'_'+str(m))==sol[ii].lhs():
                    A[i][j][m]=sol[ii].rhs()
                if var('bb_'+str(i)+'_'+str(j)+'_'+str(m))==sol[ii].lhs():
                    B[i][j][m]=sol[ii].rhs()
```

In [687]:
```
print("A[2,0,2]",A[2][0][2])
print("A[2,2,2]",A[2][2][2])
print("B[2,2,0]",B[2][2][0])
print("B[2,2,2]",B[2][2][2])
print("C[2]",C[2],"D[2]",D[2])
```

```
('A[2,0,2]', 1/16*(3*q^2 + 1)/q^(3/2))
('A[2,2,2]', 3/16*(q^4 - 1)/q^(7/2))
('B[2,2,0]', 1/8*(q^2 + 1)/q)
('B[2,2,2]', -1/8*(q^2 - 3)/q^3)
('C[2]', 0, 'D[2]', -1/8*(q^2 - 1)/q)
```

In [688]:
```
################################      verify lateral - O2          ########################
```

In [689]:
```
#eq_aa20=
expand((f_eta(0,0,2)-f_eta(pi,0,2)-2*eps)/(2*eps^2))
```

Out[689]: 0

In [690]:
```
#eq_aa21=
expand((f_eta(0,0,2)-f_eta(0,pi,2)-2*eps)/(2*eps^2))
```

Out[690]: 0

In [691]:
```
##################################### verify kinematic BC - o2 ##########################
```

In [692]:
```
#eq_aa1=
expr=f_eta_t(x,t,2)+f_u(x,t,1)*f_eta_x(x,t,1)-f_w(x,t,2)
```

In [693]:
```
expr1=expr.expand().coefficients(eps)
```

In [694]:
```
#suppress to save memory
#expr1
for i in range(len(expr1)):
    print(expr1[i][1])
```

```
2
3
```

In [695]:
```
expr2=expr1[0][0]
print(expr2.simplify_full())
```

```
0
```

In [696]:
```
################################### verify dynamic BC - o2 ##########################
```

In [697]:
```
eq_aa30=f_phi_t(x,t,2)+1/2*(f_u(x,t,2)^2+f_w(x,t,2)^2)+f_eta(x,t,2)-fB(2)
```

In [698]:
```
expr1=eq_aa30.expand().coefficients(eps)
```

In [699]:
```
for i in range(len(expr1)):
    print(expr1[i][1])
```

```
2
3
4
5
6
```

In [700]:
```
expr2=expr1[0][0]
print(expr2.simplify_full())
```

```
0
```

In [701]:
```
#############################################################################################
#############################################################################################
#############################################################################################
##############################          order 3                  ############################
#############################################################################################
#############################################################################################
#############################################################################################
```

In [702]:
```
#BCs
```

In [703]:
```
#######################         wave height constraint order 3          ###############
```

In [704]:
```
eq_aa20=expand((f_eta(0,0,3)-f_eta(pi,0,3)-2*eps)/(2*eps^3))
```

In [705]:
```
eq_aa20
```

Out[705]:
```
bb_3_1_1 + bb_3_1_3 + bb_3_3_1 + bb_3_3_3
```

In [706]:
```
eq_aa21=expand((f_eta(0,0,3)-f_eta(0,pi,3)-2*eps)/(2*eps^3))
```

In [707]:
```
eq_aa21
```

Out[707]:
```
bb_3_1_1 + bb_3_1_3 + bb_3_3_1 + bb_3_3_3
```

In [708]:
```
#######################                kinematic BC order 3          #################
```

In [709]:
```
eq_aa1=f_eta_t(x,t,3)+f_u(x,t,2)*f_eta_x(x,t,2)-f_w(x,t,3)
```

In [710]:
```
eq_aa2=eq_aa1.expand().coefficients(eps)
```

In [711]:
```
#suppress to save memory
#eq_aa2
for i in range(len(eq_aa2)):
    print(eq_aa2[i][1])
```

```
2
3
4
5
```

In [712]:
```
eq_aa2[0][0].simplify_full()
```

Out[712]:
```
0
```

In [713]:
```
eq_aa10=eq_aa2[1][0]
```

In [714]:
```
#eq_aa10
```

In [715]:
```
#obtain coefficients of double fourier series of the form a_kl*sin(kx)*sin(lt) using int
counter_=0
temp=0
eq_ct_11=[0 for i in range(100)]
for n1 in range(3+1):
    for n2 in range (3+1):
        if n1!=0 and n2!=0:
            temp=expand(integral(integral(eq_aa10*sin(n1*x)*sin(n2*t)/pi^2,x,0,2*pi),t,0
        if temp!=0:
            eq_ct_11[counter_]=temp
            counter_=counter_+1
        if n1!=0:
            temp=expand(integral(integral(eq_aa10*sin(n1*x)*cos(n2*t)/pi^2,x,0,2*pi),t,0
        if temp!=0:
            eq_ct_11[counter_]=temp
            counter_=counter_+1
        if n2!=0:
            temp=expand(integral(integral(eq_aa10*cos(n1*x)*sin(n2*t)/pi^2,x,0,2*pi),t,0
        if temp!=0:
            eq_ct_11[counter_]=temp
            counter_=counter_+1
        temp=expand(integral(integral(eq_aa10*cos(n1*x)*cos(n2*t)/pi^2,x,0,2*pi),t,0,2*p
        if temp!=0:
            eq_ct_11[counter_]=temp
            counter_=counter_+1
eq_ct_12=[0 for i in range(counter_)]
for i in range(counter_):
    eq_ct_12[i]=eq_ct_11[i]
```

In [716]:
```
eq_ct_12
```

Out[716]:
```
[-aa_3_1_1*q - bb_3_1_1*sqrt(q) - cc_3 - 1/8*sqrt(q) - 3/32/q^(3/2) + 3/16/q^(7/2),
 -aa_3_1_3*q - 3*bb_3_1_3*sqrt(q) - 1/16*sqrt(q) + 1/32/q^(3/2),
 -3*aa_3_3_1*q^7/(3*q^6 + q^4) - 3*bb_3_3_1*q^(13/2)/(3*q^6 + q^4) - 9*aa_3_3_1*q^5/(3
*q^6 + q^4) - bb_3_3_1*q^(9/2)/(3*q^6 + q^4) + 27/32*q^(9/2)/(3*q^6 + q^4) + 9/32*q^
(5/2)/(3*q^6 + q^4),
 -3*aa_3_3_3*q^7/(3*q^6 + q^4) - 9*bb_3_3_3*q^(13/2)/(3*q^6 + q^4) - 9/16*q^(13/2)/(3*
q^6 + q^4) - 9*aa_3_3_3*q^5/(3*q^6 + q^4) - 3*bb_3_3_3*q^(9/2)/(3*q^6 + q^4) - 15/32*q
^(9/2)/(3*q^6 + q^4) + 51/32*q^(5/2)/(3*q^6 + q^4) + 9/16*sqrt(q)/(3*q^6 + q^4)]
```

In [717]:
```
######################                dynamic BC order 3          ####################
```

In [718]:
```
eq_aa30=f_phi_t(x,t,3)+1/2*(f_u(x,t,2)^2+f_w(x,t,2)^2)+f_eta(x,t,3)-fB(3)
```

In [719]:
```
eq_aa31=eq_aa30.simplify_full().coefficients(eps)
```

In [720]:
```
#suppress to save memory
#eq_aa31
```

In [721]:
```
for i in range(len(eq_aa31)):
    print(eq_aa31[i][1])
```

```
2
3
4
5
6
7
```

In [722]:
```
eq_aa31[0][0].simplify_full()
```

Out[722]:
```
0
```

In [723]:
```
eq_aa32=eq_aa31[1][0].simplify_full().trig_reduce()
```

In [724]:
```
eq_aa32
```

Out[724]:
```
1/64*(32*(3*aa_3_3_3*q^4*cos(3*t + 3*x) + 3*aa_3_1_3*q^4*cos(3*t + x) + aa_3_3_1*q^4*c
os(t + 3*x) + aa_3_1_1*q^4*cos(t + x) + aa_3_3_1*q^4*cos(-t + 3*x) + aa_3_1_1*q^4*cos
(-t + x) + 3*aa_3_3_3*q^4*cos(-3*t + 3*x) + 3*aa_3_1_3*q^4*cos(-3*t + x))*q^5 - (64*dd
_3*q^4 - ((32*bb_3_3_3 + 15)*q^4 - 21*q^2 + 3)*cos(3*t + 3*x) - ((32*bb_3_1_3 + 11)*q^
4 - 21*q^2 - 3)*cos(3*t + x) + (2*q^6 - (32*bb_3_3_1 + 5)*q^4 + 9*q^2 + 3)*cos(t + 3*
x) + (2*q^6 - (32*bb_3_1_1 + 1)*q^4 + 9*q^2 - 3)*cos(t + x) + (2*q^6 - (32*bb_3_3_1 +
5)*q^4 + 9*q^2 + 3)*cos(-t + 3*x) + (2*q^6 - (32*bb_3_1_1 + 1)*q^4 + 9*q^2 - 3)*cos(-t
+ x) - ((32*bb_3_3_3 + 15)*q^4 - 21*q^2 + 3)*cos(-3*t + 3*x) - ((32*bb_3_1_3 + 11)*q^4
- 21*q^2 - 3)*cos(-3*t + x))*q^(9/2) - 32*(cc_3*q^4*cos(t + x) + cc_3*q^4*cos(-t + x))
*q^4)/q^(17/2)
```

In [725]:
```
#obtain coefficients of double fourier series of the form a_kl*sin(kx)*sin(lt) using int
counter_=0
temp=0
eq_aa33=[0 for i in range(40)]
for n1 in range(3+1):
    for n2 in range (3+1):
        if n1!=0 and n2!=0:
            temp=integral(integral(eq_aa32*sin(n1*x)*sin(n2*t)/pi^2,x,0,2*pi),t,0,2*pi)
        if temp!=0:
            eq_aa33[counter_]=temp.simplify_full()
            counter_=counter_+1
        if n1!=0:
            temp=integral(integral(eq_aa32*sin(n1*x)*cos(n2*t)/pi^2,x,0,2*pi),t,0,2*pi)
        if temp!=0:
            eq_aa33[counter_]=temp.simplify_full()
            counter_=counter_+1
        if n2!=0:
            temp=integral(integral(eq_aa32*cos(n1*x)*sin(n2*t)/pi^2,x,0,2*pi),t,0,2*pi)
        if temp!=0:
            eq_aa33[counter_]=temp.simplify_full()
            counter_=counter_+1
        temp=integral(integral(eq_aa32*cos(n1*x)*cos(n2*t)/pi^2,x,0,2*pi),t,0,2*pi)
        if temp!=0:
            eq_aa33[counter_]=temp.simplify_full()
            counter_=counter_+1
eq_aa_34=[0 for i in range(counter_)]
for i in range(counter_):
    eq_aa_34[i]=eq_aa33[i]
```

In [726]:
```
eq_aa_34
```

Out[726]:
```
[-4*dd_3,
 -4*dd_3,
 -4*dd_3,
 1/32*(32*aa_3_1_1*q^5 - 32*cc_3*q^4 - (2*q^6 - (32*bb_3_1_1 + 1)*q^4 + 9*q^2 - 3)*sqr
t(q))/q^(9/2),
 1/32*(96*aa_3_1_3*q^5 + ((32*bb_3_1_3 + 11)*q^4 - 21*q^2 - 3)*sqrt(q))/q^(9/2),
 1/32*(96*aa_3_1_3*q^5 + ((32*bb_3_1_3 + 11)*q^4 - 21*q^2 - 3)*sqrt(q))/q^(9/2),
 1/32*(32*aa_3_3_1*q^5 - (2*q^6 - (32*bb_3_3_1 + 5)*q^4 + 9*q^2 + 3)*sqrt(q))/q^(9/2),
 1/32*(96*aa_3_3_3*q^5 + ((32*bb_3_3_3 + 15)*q^4 - 21*q^2 + 3)*sqrt(q))/q^(9/2)]
```

In [727]:
```
######################                combine BCs-03          ########################
```

In [728]:
```
counter_=0
eq_order=[0 for i in range(2+len(eq_ct_12)+len(eq_aa_34))]
eq_order[0]=eq_aa20
eq_order[1]=eq_aa21
for i in range(len(eq_ct_12)):
    eq_order[i+2]=eq_ct_12[i]
for i in range(len(eq_aa_34)):
    eq_order[i+2+len(eq_ct_12)]=eq_aa_34[i]
```

In [729]:
```
eq_order
```

Out[729]:
```
[bb_3_1_1 + bb_3_1_3 + bb_3_3_1 + bb_3_3_3,
 bb_3_1_1 + bb_3_1_3 + bb_3_3_1 + bb_3_3_3,
 -aa_3_1_1*q - bb_3_1_1*sqrt(q) - cc_3 - 1/8*sqrt(q) - 3/32/q^(3/2) + 3/16/q^(7/2),
 -aa_3_1_3*q - 3*bb_3_1_3*sqrt(q) - 1/16*sqrt(q) + 1/32/q^(3/2),
 -3*aa_3_3_1*q^7/(3*q^6 + q^4) - 3*bb_3_3_1*q^(13/2)/(3*q^6 + q^4) - 9*aa_3_3_1*q^5/(3
*q^6 + q^4) - bb_3_3_1*q^(9/2)/(3*q^6 + q^4) + 27/32*q^(9/2)/(3*q^6 + q^4) + 9/32*q^
(5/2)/(3*q^6 + q^4),
 -3*aa_3_3_3*q^7/(3*q^6 + q^4) - 9*bb_3_3_3*q^(13/2)/(3*q^6 + q^4) - 9/16*q^(13/2)/(3*
q^6 + q^4) - 9*aa_3_3_3*q^5/(3*q^6 + q^4) - 3*bb_3_3_3*q^(9/2)/(3*q^6 + q^4) - 15/32*q
^(9/2)/(3*q^6 + q^4) + 51/32*q^(5/2)/(3*q^6 + q^4) + 9/16*sqrt(q)/(3*q^6 + q^4),
 -4*dd_3,
 -4*dd_3,
 -4*dd_3,
 1/32*(32*aa_3_1_1*q^5 - 32*cc_3*q^4 - (2*q^6 - (32*bb_3_1_1 + 1)*q^4 + 9*q^2 - 3)*sqr
t(q))/q^(9/2),
 1/32*(96*aa_3_1_3*q^5 + ((32*bb_3_1_3 + 11)*q^4 - 21*q^2 - 3)*sqrt(q))/q^(9/2),
 1/32*(96*aa_3_1_3*q^5 + ((32*bb_3_1_3 + 11)*q^4 - 21*q^2 - 3)*sqrt(q))/q^(9/2),
 1/32*(32*aa_3_3_1*q^5 - (2*q^6 - (32*bb_3_3_1 + 5)*q^4 + 9*q^2 + 3)*sqrt(q))/q^(9/2),
 1/32*(96*aa_3_3_3*q^5 + ((32*bb_3_3_3 + 15)*q^4 - 21*q^2 + 3)*sqrt(q))/q^(9/2)]
```

In [730]:
```
######################                list of vars-03         ########################
```

In [731]:
```
#flattening the equations to remove the list property to be able to extract variable nam
eps1=var('eps1')
temp=0
for i in range(len(eq_order)):
    temp=temp+eq_order[i]*eps1^i
```

In [732]:
```
list_var=temp.variables()
```

In [733]:
```
#list_var
```

In [734]:
```
list_var2=[0 for i in range(len(list_var)-2)]
```

In [735]:
```
#copy variable list, leave out eps1 & q
ii=var('ii')
ii=0
for i in range(len(list_var)):
    if list_var[i]!=eps1 and list_var[i]!=q:
        list_var2[ii]=list_var[i]
        ii=ii+1
```

In [736]:
```
list_var2
```

Out[736]:
```
[aa_3_1_1,
 aa_3_1_3,
 aa_3_3_1,
 aa_3_3_3,
 bb_3_1_1,
 bb_3_1_3,
 bb_3_3_1,
 bb_3_3_3,
 cc_3,
 dd_3]
```

In [737]:
```
len(eq_order),len(list_var2)
```

Out[737]:
```
(14, 10)
```

In [ ]:

In [738]:
```
sol=solve(eq_order,list_var2)[0][:]
```

In [739]:
```
sol
```

Out[739]:
```
[aa_3_1_1 == 1/256*(6*q^(21/2) + 11*q^(17/2) - 63*q^(13/2) + 96*q^(9/2) + 27*q^(5/2) +
27*sqrt(q))/q^7,
 aa_3_1_3 == -1/256*(31*q^(9/2) - 62*q^(5/2) - 9*sqrt(q))/q^5,
 aa_3_3_1 == -1/256*(6*q^(17/2) - 13*q^(13/2) - 5*q^(9/2) + 9*q^(5/2) + 3*sqrt(q))/q^
5,
 aa_3_3_3 == -1/256*(39*q^(13/2) - 53*q^(9/2) + 5*q^(5/2) + 9*sqrt(q))/q^7,
 bb_3_1_1 == -1/256*(6*q^10 + 3*q^8 - 43*q^6 + 72*q^4 + 15*q^2 + 27)/q^6,
 bb_3_1_3 == 1/256*(5*q^4 - 18*q^2 - 3)/q^4,
 bb_3_3_1 == 3/256*(2*q^8 + q^6 - 15*q^4 + 27*q^2 + 9)/q^4,
 bb_3_3_3 == -3/256*(q^6 - 3*q^4 + 3*q^2 - 9)/q^6,
 cc_3 == -1/64*(2*q^(13/2) + 3*q^(9/2) + 12*q^(5/2) - 9*sqrt(q))/q^4,
 dd_3 == 0]
```

In [740]:
```
########################          var substitution-03          ############################
```

In [741]:
```
#from list of solve results, compare left hand sides of expressions to variable names aa
#if there was a match, assign the right hand side of solve results to A, B, C, D
#therefore aa, bb, cc and dd are intact
for ii in range(len(sol)):
    for i in range(1,6):
        if var('cc_'+str(i))==sol[ii].lhs():
            C[i]=sol[ii].rhs()
        if var('dd_'+str(i))==sol[ii].lhs():
            D[i]=sol[ii].rhs()
        for j in range(6):
            for m in range(6):
                if var('aa_'+str(i)+'_'+str(j)+'_'+str(m))==sol[ii].lhs():
                    A[i][j][m]=sol[ii].rhs()
                if var('bb_'+str(i)+'_'+str(j)+'_'+str(m))==sol[ii].lhs():
                    B[i][j][m]=sol[ii].rhs()
```

In [742]:
```
print("A[3,1,1]",A[3][1][1])
print("A[3,1,3]",A[3][1][3])
print("A[3,3,1]",A[3][3][1])
print("A[3,3,3]",A[3][3][3])
print("B[3,1,1]",B[3][1][1])
print("B[3,1,3]",B[3][1][3])
print("B[3,3,1]",B[3][3][1])
print("B[3,3,3]",B[3][3][3])
print("C[3]",C[3],"D[3]",D[3])
```

```
('A[3,1,1]', 1/256*(6*q^(21/2) + 11*q^(17/2) - 63*q^(13/2) + 96*q^(9/2) + 27*q^(5/2) +
27*sqrt(q))/q^7)
('A[3,1,3]', -1/256*(31*q^(9/2) - 62*q^(5/2) - 9*sqrt(q))/q^5)
('A[3,3,1]', -1/256*(6*q^(17/2) - 13*q^(13/2) - 5*q^(9/2) + 9*q^(5/2) + 3*sqrt(q))/q^
5)
('A[3,3,3]', -1/256*(39*q^(13/2) - 53*q^(9/2) + 5*q^(5/2) + 9*sqrt(q))/q^7)
('B[3,1,1]', -1/256*(6*q^10 + 3*q^8 - 43*q^6 + 72*q^4 + 15*q^2 + 27)/q^6)
('B[3,1,3]', 1/256*(5*q^4 - 18*q^2 - 3)/q^4)
('B[3,3,1]', 3/256*(2*q^8 + q^6 - 15*q^4 + 27*q^2 + 9)/q^4)
('B[3,3,3]', -3/256*(q^6 - 3*q^4 + 3*q^2 - 9)/q^6)
('C[3]', -1/64*(2*q^(13/2) + 3*q^(9/2) + 12*q^(5/2) - 9*sqrt(q))/q^4, 'D[3]', 0)
```

In [743]:
```
################################      verify lateral - O3          ########################
```

In [744]:
```
#eq_aa20=
expand((f_eta(0,0,3)-f_eta(pi,0,3)-2*eps)/(2*eps^3))
```

Out[744]: 0

In [745]:
```
#eq_aa21=
expand((f_eta(0,0,3)-f_eta(0,pi,3)-2*eps)/(2*eps^3))
```

Out[745]: 0

In [746]:
```
################################### verify kinematic BC - o3 ##########################
```

In [747]:
```
#eq_aa1=
expr1=f_eta_t(x,t,3)+f_u(x,t,2)*f_eta_x(x,t,2)-f_w(x,t,3)
```

In [748]:
```
expr=expr1.simplify_full().coefficients(eps)
```

In [749]:
```
#suppress to save memory
#expr
```

In [750]:
```
for i in range(len(expr)):
    print(expr[i][1])
```

```
4
5
```

In [751]:
```
################################### verify dynamic BC - o3 ##########################
```

In [752]:
```
#eq_aa30=
expr1=f_phi_t(x,t,3)+1/2*(f_u(x,t,2)^2+f_w(x,t,2)^2)+f_eta(x,t,3)-fB(3)
```

In [753]:
```
expr=expr1.simplify_full().coefficients(eps)
```

In [754]:
```
#suppress to save memory
#expr
```

In [755]:
```
for i in range(len(expr)):
    print(expr[i][1])
```

```
4
5
6
7
```

In [756]:
```
################################################################################################
################################################################################################
################################################################################################
################################          order 4                    ###################
################################################################################################
################################################################################################
################################################################################################
```

In [757]:
```
#BCs
```

In [758]:
```
#######################         wave height constraint order 4          ###############
```

In [759]:
```
eq_aa20=expand((f_eta(0,0,4)-f_eta(pi,0,4)-2*eps)/(2*eps^4))
```

In [760]:
```
eq_aa20
```

Out[760]: 0

In [761]:
```
eq_aa21=expand((f_eta(0,0,4)-f_eta(0,pi,4)-2*eps)/(2*eps^4))
```

In [762]:
```
eq_aa21
```

Out[762]: 0

In [763]:
```
#######################                kinematic BC order 4          ##################
```

In [764]:
```
eq_aa1=f_eta_t(x,t,4)+f_u(x,t,3)*f_eta_x(x,t,3)-f_w(x,t,4)
```

In [765]:
```
#coefficient & coefficients produce erroneous results, therefore a new method is used
#this is based on calculating the remainder of a polynomial with respect to another poly
```

In [766]:
```
eq_aa2=eq_aa1.simplify_full().coefficients(eps)
```

In [767]:
```
for i in range(len(eq_aa2)):
    print(eq_aa2[i][1])
#this means that orders 1 to 3 are zero because they are not shown
```

```
4
5
6
7
8
```

In [768]:
```
eq_aa10=eq_aa2[0][0].trig_reduce().simplify_full()
```

In [769]:
```
#eq_aa10
```

In [770]:
```
#obtain coefficients of double fourier series of the form a_kl*sin(kx)*sin(lt) using int
counter_=0
temp=0
eq_ct_11=[0 for i in range(200)]
for n1 in range(4+1):
    for n2 in range (4+1):
        temp=expand(integral(integral(eq_aa10*sin(n1*x)*sin(n2*t)/pi^2,x,0,2*pi),t,0,2*p
        if temp!=0:
            eq_ct_11[counter_]=temp
            counter_=counter_+1
        temp=expand(integral(integral(eq_aa10*sin(n1*x)*cos(n2*t)/pi^2,x,0,2*pi),t,0,2*p
        if temp!=0:
            eq_ct_11[counter_]=temp
            counter_=counter_+1
        temp=expand(integral(integral(eq_aa10*cos(n1*x)*sin(n2*t)/pi^2,x,0,2*pi),t,0,2*p
        if temp!=0:
            eq_ct_11[counter_]=temp
            counter_=counter_+1
        temp=expand(integral(integral(eq_aa10*cos(n1*x)*cos(n2*t)/pi^2,x,0,2*pi),t,0,2*p
        if temp!=0:
            eq_ct_11[counter_]=temp
            counter_=counter_+1
eq_ct_12=[0 for i in range(counter_)]
for i in range(counter_):
    eq_ct_12[i]=eq_ct_11[i]
```

In [771]:
```
len(eq_ct_12)
```

Out[771]: 5

In [772]:
```
#eq_ct_12
```

In [773]:
```
########################                dynamic BC order 4          ####################
```

In [774]:
```
eq_aa30=f_phi_t(x,t,4)+1/2*(f_u(x,t,3)^2+f_w(x,t,3)^2)+f_eta(x,t,4)-fB(4)
```

In [775]:
```
eq_aa31=eq_aa30.simplify_full().expand().coefficients(eps)
```

In [776]:
```
for i in range(len(eq_aa31)):
    print(eq_aa31[i][1])
```

```
4
5
6
7
8
9
10
```

In [777]:
```
eq_aa32=eq_aa31[0][0].trig_reduce().simplify_full()
```

In [778]:
```
#eq_aa32
```

In [779]:
```
#obtain coefficients of double fourier series of the form a_kl*sin(kx)*sin(lt) using int
counter_=0
temp=0
eq_aa33=[0 for i in range(200)]
for n1 in range(4+1):
    for n2 in range (4+1):
        temp=expand(integral(integral(eq_aa32*sin(n1*x)*sin(n2*t)/pi^2,x,0,2*pi),t,0,2*p
        if temp!=0:
            eq_aa33[counter_]=temp
            counter_=counter_+1
        temp=expand(integral(integral(eq_aa32*sin(n1*x)*cos(n2*t)/pi^2,x,0,2*pi),t,0,2*p
        if temp!=0:
            eq_aa33[counter_]=temp
            counter_=counter_+1
        temp=expand(integral(integral(eq_aa32*cos(n1*x)*sin(n2*t)/pi^2,x,0,2*pi),t,0,2*p
        if temp!=0:
            eq_aa33[counter_]=temp
            counter_=counter_+1
        temp=expand(integral(integral(eq_aa32*cos(n1*x)*cos(n2*t)/pi^2,x,0,2*pi),t,0,2*p
        if temp!=0:
            eq_aa33[counter_]=temp
            counter_=counter_+1
eq_aa_34=[0 for i in range(counter_)]
for i in range(counter_):
    eq_aa_34[i]=eq_aa33[i]
```

In [780]:
```
len(eq_aa_34),eq_aa_34
```

Out[780]:
```
(10,
 [3/128*q^5 + 5/256*q^3 - 4*dd_4 - 7/32*q + 87/256/q + 39/256/q^3 - 9/32/q^5 - 9/256/q
^7,
  9/256*q^5 + 27/512*q^3 + 4*aa_4_0_2*sqrt(q) - 91/512*q + 43/64/q - 61/256/q^3 + 117/
512/q^5 + 27/512/q^7,
  8*aa_4_0_4*sqrt(q) - 67/512*q + 235/512/q + 185/512/q^3 - 207/512/q^5 - 9/256/q^7,
  -cc_4/sqrt(q),
  1/16*q^3 + 2*bb_4_2_0 + 9/256*q - 23/512/q + 27/512/q^3 + 63/512/q^5 + 27/512/q^7,
  -3/256*q^3 + 2*aa_4_2_2*sqrt(q) + bb_4_2_2 - 151/1536*q + 55/256/q + 5/64/q^3 - 3/6
4/q^5 - 27/512/q^7,
  4*aa_4_2_4*sqrt(q) + bb_4_2_4 - 199/1536*q + 79/1024/q + 397/1024/q^3 - 375/1024/q^5
- 27/1024/q^7,
  -3/256*q^5 - 21/512*q^3 + 2*bb_4_4_0 + 33/256*q - 1/256/q - 33/256/q^3 - 81/512/q^5
- 9/256/q^7,
  -9/512*q^5 + 9/1024*q^3 + 2*aa_4_4_2*sqrt(q) + bb_4_4_2 + 559/3072*q - 171/512/q + 2
7/128/q^3 - 195/1024/q^5 - 27/1024/q^7,
  4*aa_4_4_4*sqrt(q) + bb_4_4_4 - 197/3072*q - 1/4/q + 393/512/q^3 - 69/128/q^5 + 45/1
024/q^7])
```

In [781]:
```
########################              combine BCs-O4          ########################
```

In [782]:
```
counter_=0
eq_order=[0 for i in range(2+len(eq_ct_12)+len(eq_aa_34))]
eq_order[0]=eq_aa20
eq_order[1]=eq_aa21
for i in range(len(eq_ct_12)):
    eq_order[i+2]=eq_ct_12[i]
for i in range(len(eq_aa_34)):
    eq_order[i+2+len(eq_ct_12)]=eq_aa_34[i]
```

In [783]:
```
#eq_order
```

In [784]:
```
########################              list of vars-O4         ########################
```

In [785]:
```
#flattening the equations to remove the list property to be able to extract variable nam
eps1=var('eps1')
temp=0
for i in range(len(eq_order)):
    temp=temp+eq_order[i]*eps1^i
```

In [786]:
```
list_var=temp.variables()
```

In [ ]:
```

```

In [787]:
```
list_var2=[0 for i in range(len(list_var)-2)]
```

In [788]:
```
#copy variable list, leave out eps1 & q
ii=var('ii')
ii=0
for i in range(len(list_var)):
    if list_var[i]!=eps1 and list_var[i]!=q:
        list_var2[ii]=list_var[i]
        ii=ii+1
```

In [789]:
```
list_var2
```

Out[789]:
```
[aa_4_0_2,
 aa_4_0_4,
 aa_4_2_2,
 aa_4_2_4,
 aa_4_4_2,
 aa_4_4_4,
 bb_4_2_0,
 bb_4_2_2,
 bb_4_2_4,
 bb_4_4_0,
 bb_4_4_2,
 bb_4_4_4,
 cc_4,
 dd_4]
```

In [790]:
```
len(eq_order),len(list_var2)
```

Out[790]:
```
(17, 14)
```

In [791]:
```
sol=solve(eq_order,list_var2)[0][:]
```

In [792]:
```
sol
```

Out[792]:
```
[aa_4_0_2 == -1/2048*(18*q^12 + 27*q^10 - 91*q^8 + 344*q^6 - 122*q^4 + 117*q^2 + 27)/q
^(15/2),
 aa_4_0_4 == 1/4096*(67*q^8 - 235*q^6 - 185*q^4 + 207*q^2 + 18)/q^(15/2),
 aa_4_2_2 == 1/3072*(18*q^12 + 259*q^10 - 240*q^8 - 256*q^6 + 252*q^4 + 189*q^2 + 16
2)/q^(19/2),
 aa_4_2_4 == 1/3072*(398*q^10 + 63*q^8 - 1298*q^6 + 144*q^4 + 1188*q^2 + 81)*sqrt(q)/
(4*q^10 + 3*q^8),
 aa_4_4_2 == 1/6144*(54*q^14 + 207*q^12 - 1060*q^10 - 1743*q^8 + 4502*q^6 - 207*q^4 -
648*q^2 - 81)*sqrt(q)/(q^10 + 3*q^8),
 aa_4_4_4 == 1/12288*(197*q^12 + 1732*q^10 + 1481*q^8 - 9872*q^6 + 7623*q^4 - 756*q^2
 - 405)*sqrt(q)/(q^12 + 5*q^10),
 bb_4_2_0 == -1/1024*(32*q^10 + 18*q^8 - 23*q^6 + 27*q^4 + 63*q^2 + 27)/q^7,
 bb_4_2_2 == -1/768*(54*q^10 + 45*q^8 - 68*q^6 + 90*q^4 + 54*q^2 + 81)/q^9,
 bb_4_2_4 == -1/3072*(6*q^8 + 283*q^6 - 351*q^4 + 1053*q^2 + 81)/(4*q^9 + 3*q^7),
 bb_4_4_0 == 1/1024*(6*q^12 + 21*q^10 - 66*q^8 + 2*q^6 + 66*q^4 + 81*q^2 + 18)/q^7,
 bb_4_4_2 == -1/768*(18*q^12 - 105*q^10 - 273*q^8 + 518*q^6 + 288*q^4 - 621*q^2 - 81)/
(q^9 + 3*q^7),
 bb_4_4_4 == 1/3072*(21*q^10 + q^8 - 262*q^6 + 522*q^4 + 81*q^2 + 405)/(q^11 + 5*q^9),
 cc_4 == 0,
 dd_4 == 1/1024*(6*q^12 + 5*q^10 - 56*q^8 + 87*q^6 + 39*q^4 - 72*q^2 - 9)/q^7]
```

In [793]:
```
########################          var substitution-04          ##########################
```

In [794]:
```
#from list of solve results, compare left hand sides of expressions to variable names aa
#if there was a match, assign the right hand side of solve results to A, B, C, D
#therefore aa, bb, cc and dd are intact
for ii in range(len(sol)):
    for i in range(1,6):
        if var('cc_'+str(i))==sol[ii].lhs():
            C[i]=sol[ii].rhs()
            if C[i]!=0:
                print("C",i,C[i])
        if var('dd_'+str(i))==sol[ii].lhs():
            if D[i]!=0:
                D[i]=sol[ii].rhs()
            print("D",i,D[i])
        for j in range(6):
            for m in range(6):
                if var('aa_'+str(i)+'_'+str(j)+'_'+str(m))==sol[ii].lhs():
                    A[i][j][m]=sol[ii].rhs()
                    if A[i][j][m]!=0:
                        print("A",i,j,m,A[i][j][m])
                if var('bb_'+str(i)+'_'+str(j)+'_'+str(m))==sol[ii].lhs():
                    B[i][j][m]=sol[ii].rhs()
                    if B[i][j][m]!=0:
                        print("B",i,j,m,B[i][j][m])
```

```
('A', 4, 0, 2, -1/2048*(18*q^12 + 27*q^10 - 91*q^8 + 344*q^6 - 122*q^4 + 117*q^2 + 2
7)/q^(15/2))
('A', 4, 0, 4, 1/4096*(67*q^8 - 235*q^6 - 185*q^4 + 207*q^2 + 18)/q^(15/2))
('A', 4, 2, 2, 1/3072*(18*q^12 + 259*q^10 - 240*q^8 - 256*q^6 + 252*q^4 + 189*q^2 + 16
2)/q^(19/2))
('A', 4, 2, 4, 1/3072*(398*q^10 + 63*q^8 - 1298*q^6 + 144*q^4 + 1188*q^2 + 81)*sqrt
(q)/(4*q^10 + 3*q^8))
('A', 4, 4, 2, 1/6144*(54*q^14 + 207*q^12 - 1060*q^10 - 1743*q^8 + 4502*q^6 - 207*q^4
- 648*q^2 - 81)*sqrt(q)/(q^10 + 3*q^8))
('A', 4, 4, 4, 1/12288*(197*q^12 + 1732*q^10 + 1481*q^8 - 9872*q^6 + 7623*q^4 - 756*q^
2 - 405)*sqrt(q)/(q^12 + 5*q^10))
('B', 4, 2, 0, -1/1024*(32*q^10 + 18*q^8 - 23*q^6 + 27*q^4 + 63*q^2 + 27)/q^7)
('B', 4, 2, 2, -1/768*(54*q^10 + 45*q^8 - 68*q^6 + 90*q^4 + 54*q^2 + 81)/q^9)
('B', 4, 2, 4, -1/3072*(6*q^8 + 283*q^6 - 351*q^4 + 1053*q^2 + 81)/(4*q^9 + 3*q^7))
('B', 4, 4, 0, 1/1024*(6*q^12 + 21*q^10 - 66*q^8 + 2*q^6 + 66*q^4 + 81*q^2 + 18)/q^7)
('B', 4, 4, 2, -1/768*(18*q^12 - 105*q^10 - 273*q^8 + 518*q^6 + 288*q^4 - 621*q^2 - 8
1)/(q^9 + 3*q^7))
('B', 4, 4, 4, 1/3072*(21*q^10 + q^8 - 262*q^6 + 522*q^4 + 81*q^2 + 405)/(q^11 + 5*q^
9))
('D', 4, 1/1024*(6*q^12 + 5*q^10 - 56*q^8 + 87*q^6 + 39*q^4 - 72*q^2 - 9)/q^7)
```

In [795]:
```
################################      verify lateral - O4          ########################
```

In [796]:
```
#eq_aa20=
expand((f_eta(0,0,4)-f_eta(pi,0,4)-2*eps)/(2*eps^4))
```

Out[796]: 0

In [797]:
```
#eq_aa21=
expand((f_eta(0,0,4)-f_eta(0,pi,4)-2*eps)/(2*eps^4))
```

Out[797]: 0

In [798]:
```
################################### verify kinematic BC - o4 ##########################
```

In [799]:
```
expr=(f_eta_t(x,t,4)+f_u(x,t,3)*f_eta_x(x,t,3)-f_w(x,t,4)).simplify_full().coefficients(
for i in range(len(expr)):
    print(expr[i][1])
```

```
5
6
7
8
```

In [800]:
```
################################### verify dynamic BC - o4 ############################
```

In [801]:
```
expr=(f_phi_t(x,t,4)+1/2*(f_u(x,t,3)^2+f_w(x,t,3)^2)+f_eta(x,t,4)-fB(4)).simplify_full()
for i in range(len(expr)):
    print(expr[i][1])
```

```
5
6
7
8
9
10
```

In [802]:
```
#check point, about 20 minutes
```

In [803]:
```
#check point
```

In [804]:
```
#######################################################################################
#######################################################################################
#######################################################################################
##############################          order 5                  ######################
#######################################################################################
#######################################################################################
#######################################################################################
```

In [805]:
```
#BCs
```

In [806]:
```
######################         wave height constraint order 5          ###############
```

In [807]:
```
eq_aa20=expand((f_eta(0,0,5)-f_eta(pi,0,5)-2*eps)/(2*eps^5))
```

In [808]:
```
eq_aa20
```

Out[808]:
```
bb_5_1_1 + bb_5_1_3 + bb_5_1_5 + bb_5_3_1 + bb_5_3_3 + bb_5_3_5 + bb_5_5_1 + bb_5_5_3
+ bb_5_5_5
```

In [809]:
```
eq_aa21=expand((f_eta(0,0,5)-f_eta(0,pi,5)-2*eps)/(2*eps^5))
```

In [810]:
```
eq_aa21
```

Out[810]:
```
bb_5_1_1 + bb_5_1_3 + bb_5_1_5 + bb_5_3_1 + bb_5_3_3 + bb_5_3_5 + bb_5_5_1 + bb_5_5_3
+ bb_5_5_5
```

In [811]:
```
######################                kinematic BC order 5          #################
```

In [812]:
```
eq_aa1=f_eta_t(x,t,5)+f_u(x,t,4)*f_eta_x(x,t,4)-f_w(x,t,5)
```

In [813]:
```
eq_aa2=eq_aa1.simplify_full().coefficients(eps)
for i in range(len(eq_aa2)):
    print(eq_aa2[i][1])
```

```
5
6
7
8
9
10
11
```

In [814]:
```
#bottleneck
eq_aa10=eq_aa2[0][0].trig_reduce().simplify()
```

In [815]:
```
eq_aa10_=eq_aa10.simplify_full()
```

In [ ]:

In [816]:
```
#obtain coefficients of double fourier series of the form a_kl*sin(kx)*sin(lt) using int
counter_=0
temp=0
eq_ct_11=[0 for i in range(300)]
for n1 in range(5+1):
    for n2 in range (5+1):
        temp=(integral(integral(eq_aa10_*sin(n1*x)*sin(n2*t)/pi^2,x,0,2*pi),t,0,2*pi)).s
        if temp!=0:
            eq_ct_11[counter_]=temp
            print(counter_,n1,n2)
            print(temp)
            print(" ")
            counter_=counter_+1
        temp=(integral(integral(eq_aa10_*sin(n1*x)*cos(n2*t)/pi^2,x,0,2*pi),t,0,2*pi)).s
        if temp!=0:
            eq_ct_11[counter_]=temp
            print(counter_,n1,n2)
            print(temp)
            print(" ")
            counter_=counter_+1
        temp=(integral(integral(eq_aa10_*cos(n1*x)*sin(n2*t)/pi^2,x,0,2*pi),t,0,2*pi)).s
        if temp!=0:
            eq_ct_11[counter_]=temp
            print(counter_,n1,n2)
            print(temp)
            print(" ")
            counter_=counter_+1
        temp=(integral(integral(eq_aa10_*cos(n1*x)*cos(n2*t)/pi^2,x,0,2*pi),t,0,2*pi)).s
        if temp!=0:
            eq_ct_11[counter_]=temp
            print(counter_,n1,n2)
            print(temp)
            print(" ")
            counter_=counter_+1
eq_ct_12=[0 for i in range(counter_)]
for i in range(counter_):
    eq_ct_12[i]=eq_ct_11[i]
```

```
(0, 1, 1)
-1/49152*(49152*aa_5_1_1*q^11 + 49152*cc_5*q^10 + (36*q^16 - 360*q^14 - 951*q^12 + 3
*(16384*bb_5_1_1 + 1589)*q^10 - 2283*q^8 - 4450*q^6 + 4023*q^4 + 3051*q^2 + 1863)*sq
rt(q))/q^10

(1, 1, 3)
-1/16384*(65536*aa_5_1_3*q^11 + 49152*aa_5_1_3*q^9 - (192*q^14 - 88*q^12 - 2*(98304*
bb_5_1_3 + 925)*q^10 - 3*(49152*bb_5_1_3 - 1697)*q^8 - 1388*q^6 - 2628*q^4 - 2142*q^
2 - 675)*sqrt(q))/(4*q^10 + 3*q^8)

(2, 1, 5)
-1/24576*(98304*aa_5_1_5*q^11 + 73728*aa_5_1_5*q^9 + (24*(20480*bb_5_1_5 + 27)*q^10
+ 12*(30720*bb_5_1_5 - 79)*q^8 - 4522*q^6 + 3123*q^4 + 2376*q^2 - 405)*sqrt(q))/(4*q
^10 + 3*q^8)

(3, 3, 1)
-1/16384*(49152*aa_5_3_1*q^13 + 294912*aa_5_3_1*q^11 + 442368*aa_5_3_1*q^9 - (36*q^1
8 - 1104*q^16 - 11091*q^14 - 3*(16384*bb_5_3_1 - 2049)*q^12 - 5*(32768*bb_5_3_1 - 11
061)*q^10 - 3*(16384*bb_5_3_1 + 28871)*q^8 - 87225*q^6 - 6831*q^4 - 11745*q^2 - 510
3)*sqrt(q))/(3*q^12 + 10*q^10 + 3*q^8)

(4, 3, 3)
-3/16384*(65536*aa_5_3_3*q^19 + 770048*aa_5_3_3*q^17 + 3096576*aa_5_3_3*q^15 + 48660
48*aa_5_3_3*q^13 + 2211840*aa_5_3_3*q^11 + (864*q^22 + 8400*q^20 + 2*(98304*bb_5_3_3
+ 9221)*q^18 + (1785856*bb_5_3_3 + 4115)*q^16 + 2*(2351104*bb_5_3_3 + 82091)*q^14 +
2*(1794048*bb_5_3_3 + 248561)*q^12 + 24*(30720*bb_5_3_3 + 7883)*q^10 + 440982*q^8 +
754470*q^6 + 579474*q^4 + 289170*q^2 + 54675)*sqrt(q))/(12*q^18 + 109*q^16 + 287*q^1
```

```
4 + 219*q^12 + 45*q^10)

(5, 3, 5)
-1/8192*(98304*aa_5_3_5*q^11 + 368640*aa_5_3_5*q^9 + 221184*aa_5_3_5*q^7 + (384*(128
0*bb_5_3_5 + 9)*q^10 + 2*(266240*bb_5_3_5 + 4023)*q^8 + 3*(40960*bb_5_3_5 - 7727)*q^
6 - 69*q^4 + 37557*q^2 + 11583)*sqrt(q))/(12*q^10 + 13*q^8 + 3*q^6)

(6, 5, 1)
-1/24576*(122880*aa_5_5_1*q^15 + 1597440*aa_5_5_1*q^13 + 4300800*aa_5_5_1*q^11 + 184
3200*aa_5_5_1*q^9 - (5250*q^16 - 15*(8192*bb_5_5_1 - 4535)*q^14 - 75*(8192*bb_5_5_1
- 1449)*q^12 - (761856*bb_5_5_1 + 281695)*q^10 - (73728*bb_5_5_1 + 345785)*q^8 + 398
155*q^6 + 130905*q^4 + 66555*q^2 + 6075)*sqrt(q))/(5*q^14 + 25*q^12 + 31*q^10 + 3*q^
8)

(7, 5, 3)
-1/8192*(40960*aa_5_5_3*q^15 + 532480*aa_5_5_3*q^13 + 1433600*aa_5_5_3*q^11 + 614400
*aa_5_5_3*q^9 + (1500*q^18 + 10500*q^16 + 15*(8192*bb_5_5_3 - 1865)*q^14 + 50*(12288
*bb_5_5_3 - 3185)*q^12 + (761856*bb_5_5_3 + 47795)*q^10 + 12*(6144*bb_5_5_3 + 33205)
*q^8 - 23605*q^6 - 169830*q^4 - 68355*q^2 - 5400)*sqrt(q))/(5*q^14 + 25*q^12 + 31*q^
10 + 3*q^8)

(8, 5, 5)
-5/24576*(24576*aa_5_5_5*q^17 + 368640*aa_5_5_5*q^15 + 1351680*aa_5_5_5*q^13 + 61440
0*aa_5_5_5*q^11 + (30*(4096*bb_5_5_5 + 21)*q^16 + 15*(57344*bb_5_5_5 + 659)*q^14 + 3
2*(39168*bb_5_5_5 + 893)*q^12 + 5*(24576*bb_5_5_5 - 9295)*q^10 - 73440*q^8 + 106795*
q^6 - 27396*q^4 - 42525*q^2 - 4050)*sqrt(q))/(5*q^16 + 35*q^14 + 51*q^12 + 5*q^10)
```

In [817]:
```
len(eq_ct_12)
```

Out[817]: 9

In [818]:
```
#eq_ct_12
```

In [819]:
```
########################              dynamic BC order 5          ####################
```

In [820]:
```
eq_aa30=f_phi_t(x,t,5)+1/2*(f_u(x,t,4)^2+f_w(x,t,4)^2)+f_eta(x,t,5)-fB(5)
```

In [821]:
```
eq_aa31=eq_aa30.simplify_full().coefficients(eps)
for i in range(len(eq_aa31)):
    print(eq_aa31[i][1])
```

```
5
6
7
8
9
10
11
12
13
14
```

In [822]:
```
#eq_aa31[0][0]
A[5][5][5]
```

Out[822]: aa_5_5_5

In [823]:
```
eq_aa32=eq_aa31[0][0].trig_reduce().simplify()
```

In [824]:
```
eq_aa32_1=eq_aa32.simplify_full()
```

In [825]:
```
eq_aa32_2=eq_aa32_1.trig_reduce()
```

In [826]:
```
#obtain coefficients of double fourier series of the form a_kl*sin(kx)*sin(lt) using int
counter_=0
temp=0
eq_aa33=[0 for i in range(300)]
for n1 in range(5+1):
    for n2 in range (5+1):

        if n1!=0 or n2!=0:
            temp=integral(integral(eq_aa32_1*sin(n1*x)*sin(n2*t)/pi^2,x,0,2*pi),t,0,2*pi
        if temp!=0:
            eq_aa33[counter_]=temp.simplify_full()
            counter_=counter_+1
            print(counter_,n1,n2,1)
            print(eq_aa33[counter_-1])
            print(" ")
        if n1!=0:
            temp=integral(integral(eq_aa32_1*sin(n1*x)*cos(n2*t)/pi^2,x,0,2*pi),t,0,2*pi
        if temp!=0:
            eq_aa33[counter_]=temp.simplify_full()
            counter_=counter_+1
            print(counter_,n1,n2,2)
            print(eq_aa33[counter_-1])
            print(" ")
        if n2!=0:
            temp=integral(integral(eq_aa32_1*cos(n1*x)*sin(n2*t)/pi^2,x,0,2*pi),t,0,2*pi
        if temp!=0:
            eq_aa33[counter_]=temp.simplify_full()
            counter_=counter_+1
            print(counter_,n1,n2,3)
            print(eq_aa33[counter_-1])
            print(" ")
        if n1==0 and n2==1:
            temp=integral(integral(eq_aa32_2,x,0,2*pi)*cos(t),t,0,2*pi)
        else:
            temp=integral(integral(eq_aa32_1*cos(n1*x)*cos(n2*t)/pi^2,x,0,2*pi),t,0,2*pi
        if temp!=0:
            eq_aa33[counter_]=temp.simplify_full()
            counter_=counter_+1
            print(counter_,n1,n2,4)
            print(eq_aa33[counter_-1])
            print(" ")
eq_aa_34=[0 for i in range(counter_)]
for i in range(counter_):
    eq_aa_34[i]=eq_aa33[i]
```

```
(1, 0, 0, 4)
-4*dd_5

(2, 1, 1, 4)
-1/49152*(36*q^16 - 696*q^14 - 3135*q^12 - 3*(16384*bb_5_1_1 + 1187)*q^10 + 3957*q^8
- 1418*q^6 - 5697*q^4 + 27*q^2 - 49152*(aa_5_1_1*q^10 - cc_5*q^9)*sqrt(q) + 567)/q^1
0

(3, 1, 3, 4)
-1/49152*(576*q^16 - 17160*q^14 - 6*(32768*bb_5_1_3 + 8473)*q^12 - (147456*bb_5_1_3
- 529)*q^10 - 17054*q^8 - 70968*q^6 - 34830*q^4 - 51597*q^2 - 147456*(4*aa_5_1_3*q^1
2 + 3*aa_5_1_3*q^10)*sqrt(q) - 5346)/(4*q^12 + 3*q^10)

(4, 1, 5, 4)
1/24576*(12*(8192*bb_5_1_5 + 285)*q^12 + 12*(6144*bb_5_1_5 + 37)*q^10 - 20267*q^8 +
65043*q^6 - 594*q^4 - 14661*q^2 + 122880*(4*aa_5_1_5*q^12 + 3*aa_5_1_5*q^10)*sqrt(q)
- 729)/(4*q^12 + 3*q^10)

(5, 3, 1, 4)
-1/16384*(12*q^18 - 444*q^16 + 407*q^14 - 2*(8192*bb_5_3_1 + 745)*q^12 - 3*(16384*bb
```

```
_5_3_1 + 3955)*q^10 + 9670*q^8 - 8863*q^6 - 2178*q^4 - 8235*q^2 - 16384*(aa_5_3_1*q^
12 + 3*aa_5_3_1*q^10)*sqrt(q) - 1782)/(q^12 + 3*q^10)

(6, 3, 3, 4)
1/16384*(3888*q^20 + 10116*q^18 + 4*(16384*bb_5_3_3 - 22155)*q^16 + (573440*bb_5_3_3
- 150909)*q^14 + 3*(458752*bb_5_3_3 + 114433)*q^12 + 9*(81920*bb_5_3_3 + 27899)*q^10
+ 46653*q^8 + 443349*q^6 + 247617*q^4 + 49329*q^2 + 49152*(4*aa_5_3_3*q^16 + 35*aa_5
_3_3*q^14 + 84*aa_5_3_3*q^12 + 45*aa_5_3_3*q^10)*sqrt(q) - 32805)/(4*q^16 + 35*q^14
+ 84*q^12 + 45*q^10)

(7, 3, 5, 4)
1/8192*(64*(512*bb_5_3_5 + 85)*q^14 + (188416*bb_5_3_5 + 24153)*q^12 + 6*(20480*bb_5
_3_5 - 6609)*q^10 - 65737*q^8 + 101861*q^6 + 1494*q^4 - 20547*q^2 + 40960*(4*aa_5_3_
5*q^14 + 23*aa_5_3_5*q^12 + 15*aa_5_3_5*q^10)*sqrt(q) - 810)/(4*q^14 + 23*q^12 + 15*
q^10)

(8, 5, 1, 4)
-1/24576*(108*q^18 - 432*q^16 + 1095*q^14 - 12*(2048*bb_5_5_1 - 889)*q^12 - 24*(3072
*bb_5_5_1 + 247)*q^10 - 31571*q^8 + 19647*q^6 + 9126*q^4 + 6318*q^2 - 24576*(aa_5_5_
1*q^12 + 3*aa_5_5_1*q^10)*sqrt(q) + 729)/(q^12 + 3*q^10)

(9, 5, 3, 4)
1/24576*(2358*q^18 + 7563*q^16 + 48*(512*bb_5_5_3 - 945)*q^14 + 2*(98304*bb_5_5_3 -
45605)*q^12 + 3*(122880*bb_5_5_3 + 60683)*q^10 + 133225*q^8 - 178194*q^6 + 27540*q^4
- 34101*q^2 + 73728*(aa_5_5_3*q^14 + 8*aa_5_5_3*q^12 + 15*aa_5_5_3*q^10)*sqrt(q) - 2
430)/(q^14 + 8*q^12 + 15*q^10)

(10, 5, 5, 4)
1/24576*(3*(8192*bb_5_5_5 + 1125)*q^12 + 15*(8192*bb_5_5_5 + 995)*q^10 - 26095*q^8 -
32960*q^6 + 73395*q^4 - 37665*q^2 + 122880*(aa_5_5_5*q^12 + 5*aa_5_5_5*q^10)*sqrt(q)
+ 2025)/(q^12 + 5*q^10)
```

In [827]:
```
len(eq_aa_34)
```

Out[827]: 10

In [828]:
```
eq_aa_34
```

Out[828]:
```
[-4*dd_5,
 -1/49152*(36*q^16 - 696*q^14 - 3135*q^12 - 3*(16384*bb_5_1_1 + 1187)*q^10 + 3957*q^8
- 1418*q^6 - 5697*q^4 + 27*q^2 - 49152*(aa_5_1_1*q^10 - cc_5*q^9)*sqrt(q) + 567)/q^10,
 -1/49152*(576*q^16 - 17160*q^14 - 6*(32768*bb_5_1_3 + 8473)*q^12 - (147456*bb_5_1_3 -
529)*q^10 - 17054*q^8 - 70968*q^6 - 34830*q^4 - 51597*q^2 - 147456*(4*aa_5_1_3*q^12 +
3*aa_5_1_3*q^10)*sqrt(q) - 5346)/(4*q^12 + 3*q^10),
 1/24576*(12*(8192*bb_5_1_5 + 285)*q^12 + 12*(6144*bb_5_1_5 + 37)*q^10 - 20267*q^8 + 6
5043*q^6 - 594*q^4 - 14661*q^2 + 122880*(4*aa_5_1_5*q^12 + 3*aa_5_1_5*q^10)*sqrt(q) -
729)/(4*q^12 + 3*q^10),
 -1/16384*(12*q^18 - 444*q^16 + 407*q^14 - 2*(8192*bb_5_3_1 + 745)*q^12 - 3*(16384*bb_
5_3_1 + 3955)*q^10 + 9670*q^8 - 8863*q^6 - 2178*q^4 - 8235*q^2 - 16384*(aa_5_3_1*q^12
+ 3*aa_5_3_1*q^10)*sqrt(q) - 1782)/(q^12 + 3*q^10),
 1/16384*(3888*q^20 + 10116*q^18 + 4*(16384*bb_5_3_3 - 22155)*q^16 + (573440*bb_5_3_3
- 150909)*q^14 + 3*(458752*bb_5_3_3 + 114433)*q^12 + 9*(81920*bb_5_3_3 + 27899)*q^10 +
46653*q^8 + 443349*q^6 + 247617*q^4 + 49329*q^2 + 49152*(4*aa_5_3_3*q^16 + 35*aa_5_3_3
*q^14 + 84*aa_5_3_3*q^12 + 45*aa_5_3_3*q^10)*sqrt(q) - 32805)/(4*q^16 + 35*q^14 + 84*q
^12 + 45*q^10),
 1/8192*(64*(512*bb_5_3_5 + 85)*q^14 + (188416*bb_5_3_5 + 24153)*q^12 + 6*(20480*bb_5_
3_5 - 6609)*q^10 - 65737*q^8 + 101861*q^6 + 1494*q^4 - 20547*q^2 + 40960*(4*aa_5_3_5*q
^14 + 23*aa_5_3_5*q^12 + 15*aa_5_3_5*q^10)*sqrt(q) - 810)/(4*q^14 + 23*q^12 + 15*q^1
0),
 -1/24576*(108*q^18 - 432*q^16 + 1095*q^14 - 12*(2048*bb_5_5_1 - 889)*q^12 - 24*(3072*
bb_5_5_1 + 247)*q^10 - 31571*q^8 + 19647*q^6 + 9126*q^4 + 6318*q^2 - 24576*(aa_5_5_1*q
^12 + 3*aa_5_5_1*q^10)*sqrt(q) + 729)/(q^12 + 3*q^10),
 1/24576*(2358*q^18 + 7563*q^16 + 48*(512*bb_5_5_3 - 945)*q^14 + 2*(98304*bb_5_5_3 - 4
5605)*q^12 + 3*(122880*bb_5_5_3 + 60683)*q^10 + 133225*q^8 - 178194*q^6 + 27540*q^4 -
34101*q^2 + 73728*(aa_5_5_3*q^14 + 8*aa_5_5_3*q^12 + 15*aa_5_5_3*q^10)*sqrt(q) - 243
0)/(q^14 + 8*q^12 + 15*q^10),
 1/24576*(3*(8192*bb_5_5_5 + 1125)*q^12 + 15*(8192*bb_5_5_5 + 995)*q^10 - 26095*q^8 -
32960*q^6 + 73395*q^4 - 37665*q^2 + 122880*(aa_5_5_5*q^12 + 5*aa_5_5_5*q^10)*sqrt(q) +
2025)/(q^12 + 5*q^10)]
```

In [829]:
```
########################                combine BCs-O5          ########################
```

In [830]:
```
counter_=0
eq_order=[0 for i in range(len(eq_ct_12)+len(eq_aa_34)+2)]
eq_order[0]=eq_aa20
eq_order[0]=eq_aa21
for i in range(len(eq_ct_12)):
    eq_order[i+2]=eq_ct_12[i]
for i in range(len(eq_aa_34)):
    eq_order[i+2+len(eq_ct_12)]=eq_aa_34[i]
```

In [831]:
```
#eq_order
len(eq_order)
```

Out[831]:
```
21
```

In [ ]:
```

```

In [832]:
```
########################                list of vars-O5         ########################
```

In [833]:
```
#flattening the equations to remove the list property to be able to extract variable nam
eps1=var('eps1')
temp=0
for i in range(len(eq_order)):
    temp=temp+eq_order[i]*eps1^i
```

In [834]:
```
list_var=temp.variables()
```

In [835]:
```
len(list_var)
```

Out[835]: 22

In [836]:
```
list_var2=[0 for i in range(len(list_var)-2)]
```

In [837]:
```
#copy variable list, leave out eps1 & q
ii=var('ii')
ii=0
for i in range(len(list_var)):
    if list_var[i]!=eps1 and list_var[i]!=q:
        list_var2[ii]=list_var[i]
        ii=ii+1
```

In [838]:
```
list_var2,len(list_var2),len(eq_order)
```

Out[838]:
```
([aa_5_1_1,
  aa_5_1_3,
  aa_5_1_5,
  aa_5_3_1,
  aa_5_3_3,
  aa_5_3_5,
  aa_5_5_1,
  aa_5_5_3,
  aa_5_5_5,
  bb_5_1_1,
  bb_5_1_3,
  bb_5_1_5,
  bb_5_3_1,
  bb_5_3_3,
  bb_5_3_5,
  bb_5_5_1,
  bb_5_5_3,
  bb_5_5_5,
  cc_5,
  dd_5],
 20,
 21)
```

In [839]:
```
sol=solve(eq_order,list_var2)[0][:]
```

In [840]:
```
#from list of solve results, compare left hand sides of expressions to variable names aa
#if there was a match, assign the right hand side of solve results to A, B, C, D
#therefore aa, bb, cc and dd are intact
for ii in range(len(sol)):
    for i in range(1,6):
        if var('cc_'+str(i))==sol[ii].lhs():
            C[i]=sol[ii].rhs()
            #vars()['cc_'+str(i)]=sol[ii].rhs()
            if C[i]!=0:
                print("C",i,C[i])
        if var('dd_'+str(i))==sol[ii].lhs():
            D[i]=sol[ii].rhs()
            #vars()['dd_'+str(i)]=sol[ii].rhs()
            print("D",i,D[i])
        for j in range(6):
            for m in range(6):
                if var('aa_'+str(i)+'_'+str(j)+'_'+str(m))==sol[ii].lhs():
                    A[i][j][m]=sol[ii].rhs()
                    #vars()['aa_'+str(i)+'_'+str(j)+'_'+str(m)]=0
                    print("A",i,j,m,A[i][j][m])
                    if A[i][j][m]!=0:
                        print("A",i,j,m,A[i][j][m])
                if var('bb_'+str(i)+'_'+str(j)+'_'+str(m))==sol[ii].lhs():
                    B[i][j][m]=sol[ii].rhs()
                    #vars()['bb_'+str(i)+'_'+str(j)+'_'+str(m)]=0
                    print("B",i,j,m,B[i][j][m])
                    if B[i][j][m]!=0:
                        print("B",i,j,m,B[i][j][m])
```

```
('A', 5, 1, 1, 1/65536*(97200*q^34 + 969300*q^32 + 10312920*q^30 + 40437585*q^28 - 9
9942680*q^26 - 718547767*q^24 - 1053139982*q^22 - 468613281*q^20 - 625150182*q^18 -
1543359636*q^16 - 1317006294*q^14 - 610724583*q^12 - 538477502*q^10 - 434987253*q^8
- 119464668*q^6 + 24483303*q^4 + 17432820*q^2 + 2259900)/(2700*q^(53/2) + 24945*q^(4
9/2) + 73990*q^(45/2) + 93072*q^(41/2) + 46690*q^(37/2) + 1267*q^(33/2) - 5220*q^(2
9/2) - 900*q^(25/2)))
('A', 5, 1, 1, 1/65536*(97200*q^34 + 969300*q^32 + 10312920*q^30 + 40437585*q^28 - 9
9942680*q^26 - 718547767*q^24 - 1053139982*q^22 - 468613281*q^20 - 625150182*q^18 -
1543359636*q^16 - 1317006294*q^14 - 610724583*q^12 - 538477502*q^10 - 434987253*q^8
- 119464668*q^6 + 24483303*q^4 + 17432820*q^2 + 2259900)/(2700*q^(53/2) + 24945*q^(4
9/2) + 73990*q^(45/2) + 93072*q^(41/2) + 46690*q^(37/2) + 1267*q^(33/2) - 5220*q^(2
9/2) - 900*q^(25/2)))
('A', 5, 1, 3, 1/65536*(192*q^16 - 8536*q^14 - 24494*q^12 - 2281*q^10 - 7833*q^8 - 3
4170*q^6 - 16344*q^4 - 25461*q^2 - 2673)/(4*q^(25/2) + 3*q^(21/2)))
('A', 5, 1, 3, 1/65536*(192*q^16 - 8536*q^14 - 24494*q^12 - 2281*q^10 - 7833*q^8 - 3
4170*q^6 - 16344*q^4 - 25461*q^2 - 2673)/(4*q^(25/2) + 3*q^(21/2)))
('A', 5, 1, 5, -1/65536*(1828*q^12 + 352*q^10 - 10757*q^8 + 35788*q^6 - 594*q^4 - 81
00*q^2 - 405)/(4*q^(25/2) + 3*q^(21/2)))
('A', 5, 1, 5, -1/65536*(1828*q^12 + 352*q^10 - 10757*q^8 + 35788*q^6 - 594*q^4 - 81
00*q^2 - 405)/(4*q^(25/2) + 3*q^(21/2)))
('A', 5, 3, 1, 1/65536*(108*q^18 - 5934*q^16 + 5105*q^14 + 46195*q^12 - 51879*q^10 -
35153*q^8 + 4283*q^6 + 7569*q^4 + 4239*q^2 + 891)/(q^(25/2) + 3*q^(21/2)))
('A', 5, 3, 1, 1/65536*(108*q^18 - 5934*q^16 + 5105*q^14 + 46195*q^12 - 51879*q^10 -
35153*q^8 + 4283*q^6 + 7569*q^4 + 4239*q^2 + 891)/(q^(25/2) + 3*q^(21/2)))
('A', 5, 3, 3, -1/65536*(5400*q^22 + 12918*q^20 - 137093*q^18 - 272731*q^16 + 357403
*q^14 + 299725*q^12 + 100929*q^10 + 467859*q^8 + 215865*q^6 - 91935*q^4 - 169128*q^2
- 43740)/(4*q^(37/2) + 35*q^(33/2) + 84*q^(29/2) + 45*q^(25/2)))
('A', 5, 3, 3, -1/65536*(5400*q^22 + 12918*q^20 - 137093*q^18 - 272731*q^16 + 357403
*q^14 + 299725*q^12 + 100929*q^10 + 467859*q^8 + 215865*q^6 - 91935*q^4 - 169128*q^2
- 43740)/(4*q^(37/2) + 35*q^(33/2) + 84*q^(29/2) + 45*q^(25/2)))
('A', 5, 3, 5, -1/65536*(78144*q^16 + 364169*q^14 - 491094*q^12 - 1068351*q^10 + 116
2018*q^8 + 332347*q^6 - 358650*q^4 - 114885*q^2 - 4050)/(36*q^(33/2) + 215*q^(29/2)
+ 181*q^(25/2) + 30*q^(21/2)))
('A', 5, 3, 5, -1/65536*(78144*q^16 + 364169*q^14 - 491094*q^12 - 1068351*q^10 + 116
2018*q^8 + 332347*q^6 - 358650*q^4 - 114885*q^2 - 4050)/(36*q^(33/2) + 215*q^(29/2)
+ 181*q^(25/2) + 30*q^(21/2)))
```

```
('A', 5, 5, 1, -1/65536*(180*q^22 - 360*q^20 - 1329*q^18 - 1389*q^16 - 10180*q^14 +
25076*q^12 + 40794*q^10 - 62542*q^8 + 3864*q^6 + 3132*q^4 + 2511*q^2 + 243)/(5*q^(2
9/2) + 18*q^(25/2) + 9*q^(21/2)))
('A', 5, 5, 1, -1/65536*(180*q^22 - 360*q^20 - 1329*q^18 - 1389*q^16 - 10180*q^14 +
25076*q^12 + 40794*q^10 - 62542*q^8 + 3864*q^6 + 3132*q^4 + 2511*q^2 + 243)/(5*q^(2
9/2) + 18*q^(25/2) + 9*q^(21/2)))
('A', 5, 5, 3, -1/65536*(10290*q^22 + 43395*q^20 - 173337*q^18 - 602962*q^16 + 70124
0*q^14 + 1757970*q^12 - 1345366*q^10 - 1223160*q^8 + 844206*q^6 + 21555*q^4 - 31401*
q^2 - 2430)/(5*q^(37/2) + 45*q^(33/2) + 113*q^(29/2) + 59*q^(25/2) - 30*q^(21/2)))
('A', 5, 5, 3, -1/65536*(10290*q^22 + 43395*q^20 - 173337*q^18 - 602962*q^16 + 70124
0*q^14 + 1757970*q^12 - 1345366*q^10 - 1223160*q^8 + 844206*q^6 + 21555*q^4 - 31401*
q^2 - 2430)/(5*q^(37/2) + 45*q^(33/2) + 113*q^(29/2) + 59*q^(25/2) - 30*q^(21/2)))
('A', 5, 5, 5, -1/65536*(5415*q^16 + 32830*q^14 - 2142*q^12 - 121450*q^10 + 28240*q^
8 + 135290*q^6 - 88578*q^4 + 8370*q^2 + 2025)/(3*q^(33/2) + 20*q^(29/2) + 25*q^(25/
2)))
('A', 5, 5, 5, -1/65536*(5415*q^16 + 32830*q^14 - 2142*q^12 - 121450*q^10 + 28240*q^
8 + 135290*q^6 - 88578*q^4 + 8370*q^2 + 2025)/(3*q^(33/2) + 20*q^(29/2) + 25*q^(25/
2)))
('B', 5, 1, 1, -1/196608*(291600*q^34 + 2907900*q^32 + 32753160*q^30 + 149869395*q^2
8 - 96175800*q^26 - 1388122677*q^24 - 1816814730*q^22 - 673012947*q^20 - 2204709010*
q^18 - 4183229964*q^16 - 2019366082*q^14 - 127366501*q^12 - 1024292202*q^10 - 127229
7663*q^8 - 404176500*q^6 + 54476469*q^4 + 49965660*q^2 + 6779700)/(2700*q^26 + 24945
*q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12))
('B', 5, 1, 1, -1/196608*(291600*q^34 + 2907900*q^32 + 32753160*q^30 + 149869395*q^2
8 - 96175800*q^26 - 1388122677*q^24 - 1816814730*q^22 - 673012947*q^20 - 2204709010*
q^18 - 4183229964*q^16 - 2019366082*q^14 - 127366501*q^12 - 1024292202*q^10 - 127229
7663*q^8 - 404176500*q^6 + 54476469*q^4 + 49965660*q^2 + 6779700)/(2700*q^26 + 24945
*q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12))
('B', 5, 1, 3, 1/196608*(576*q^16 + 8184*q^14 + 17094*q^12 + 22645*q^10 + 2281*q^8 +
23658*q^6 + 7776*q^4 + 22761*q^2 + 2673)/(4*q^12 + 3*q^10))
('B', 5, 1, 3, 1/196608*(576*q^16 + 8184*q^14 + 17094*q^12 + 22645*q^10 + 2281*q^8 +
23658*q^6 + 7776*q^4 + 22761*q^2 + 2673)/(4*q^12 + 3*q^10))
('B', 5, 1, 5, 1/196608*(60*q^12 + 1728*q^10 + 781*q^8 + 16476*q^6 - 4158*q^4 - 4212
*q^2 - 243)/(4*q^12 + 3*q^10))
('B', 5, 1, 5, 1/196608*(60*q^12 + 1728*q^10 + 781*q^8 + 16476*q^6 - 4158*q^4 - 4212
*q^2 - 243)/(4*q^12 + 3*q^10))
('B', 5, 3, 1, -3/65536*(20*q^18 - 1386*q^16 + 1159*q^14 + 17385*q^12 - 1473*q^10 -
24611*q^8 + 13245*q^6 + 5427*q^4 + 12393*q^2 + 2673)/(q^12 + 3*q^10))
('B', 5, 3, 1, -3/65536*(20*q^18 - 1386*q^16 + 1159*q^14 + 17385*q^12 - 1473*q^10 -
24611*q^8 + 13245*q^6 + 5427*q^4 + 12393*q^2 + 2673)/(q^12 + 3*q^10))
('B', 5, 3, 3, 3/65536*(216*q^22 - 570*q^20 - 18933*q^18 - 71519*q^16 - 100329*q^14
- 35063*q^12 + 38725*q^10 - 123273*q^8 - 114291*q^6 - 157707*q^4 - 125388*q^2 - 4374
0)/(4*q^18 + 35*q^16 + 84*q^14 + 45*q^12))
('B', 5, 3, 3, 3/65536*(216*q^22 - 570*q^20 - 18933*q^18 - 71519*q^16 - 100329*q^14
- 35063*q^12 + 38725*q^10 - 123273*q^8 - 114291*q^6 - 157707*q^4 - 125388*q^2 - 4374
0)/(4*q^18 + 35*q^16 + 84*q^14 + 45*q^12))
('B', 5, 3, 5, -3/65536*(320*q^16 + 1737*q^14 - 4390*q^12 - 8591*q^10 + 157370*q^8 +
25203*q^6 + 112590*q^4 + 62451*q^2 + 2430)/(36*q^16 + 215*q^14 + 181*q^12 + 30*q^1
0))
('B', 5, 3, 5, -3/65536*(320*q^16 + 1737*q^14 - 4390*q^12 - 8591*q^10 + 157370*q^8 +
25203*q^6 + 112590*q^4 + 62451*q^2 + 2430)/(36*q^16 + 215*q^14 + 181*q^12 + 30*q^1
0))
('B', 5, 5, 1, 5/196608*(108*q^22 + 648*q^20 - 3735*q^18 + 5853*q^16 + 84492*q^14 +
18828*q^12 - 256546*q^10 - 31890*q^8 + 169632*q^6 + 96228*q^4 + 37665*q^2 + 3645)/(5
*q^14 + 18*q^12 + 9*q^10))
('B', 5, 5, 1, 5/196608*(108*q^22 + 648*q^20 - 3735*q^18 + 5853*q^16 + 84492*q^14 +
18828*q^12 - 256546*q^10 - 31890*q^8 + 169632*q^6 + 96228*q^4 + 37665*q^2 + 3645)/(5
*q^14 + 18*q^12 + 9*q^10))
('B', 5, 5, 3, -5/196608*(342*q^22 + 1257*q^20 + 2085*q^18 - 31430*q^16 - 390368*q^1
4 - 350282*q^12 + 1479350*q^10 + 570136*q^8 - 1001838*q^6 - 419175*q^4 + 146205*q^2
+ 12150)/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^12 - 30*q^10))
('B', 5, 5, 3, -5/196608*(342*q^22 + 1257*q^20 + 2085*q^18 - 31430*q^16 - 390368*q^1
4 - 350282*q^12 + 1479350*q^10 + 570136*q^8 - 1001838*q^6 - 419175*q^4 + 146205*q^2
+ 12150)/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^12 - 30*q^10))
('B', 5, 5, 5, 5/196608*(45*q^16 - 150*q^14 - 570*q^12 + 2618*q^10 - 3896*q^8 - 498*
q^6 + 25866*q^4 + 8910*q^2 + 6075)/(3*q^16 + 20*q^14 + 25*q^12))
```

```
('B', 5, 5, 5, 5/196608*(45*q^16 - 150*q^14 - 570*q^12 + 2618*q^10 - 3896*q^8 - 498*
q^6 + 25866*q^4 + 8910*q^2 + 6075)/(3*q^16 + 20*q^14 + 25*q^12))
('C', 5, -1/16384*(12*q^16 - 176*q^14 - 681*q^12 + 201*q^10 + 279*q^8 - 978*q^6 - 27
9*q^4 + 513*q^2 + 405)/q^(19/2))
('D', 5, 0)
```

In [841]:
```
for i in range(1,6):
    if var('cc_'+str(i))!=C[i]:
        vars()['cc_'+str(i)]=C[i]
    if var('dd_'+str(i))!=D[i]:
        vars()['dd_'+str(i)]=D[i]
    for j in range(6):
        for m in range(6):
            if var('aa_'+str(i)+'_'+str(j)+'_'+str(m))!=A[i][j][m]:
                vars()['aa_'+str(i)+'_'+str(j)+'_'+str(m)]=A[i][j][m]
            if var('bb_'+str(i)+'_'+str(j)+'_'+str(m))!=B[i][j][m]:
                vars()['bb_'+str(i)+'_'+str(j)+'_'+str(m)]=B[i][j][m]
```

In [842]:
```
A[5][4][3],aa_5_4_3,B[5][5][4],bb_5_5_4,D[3],dd_3
```

Out[842]:
```
(0, 0, 0, 0, 0, 0)
```

In [843]:
```
sol
```

Out[843]:
```
[aa_5_1_1 == 1/65536*(97200*q^34 + 969300*q^32 + 10312920*q^30 + 40437585*q^28 - 99942
680*q^26 - 718547767*q^24 - 1053139982*q^22 - 468613281*q^20 - 625150182*q^18 - 154335
9636*q^16 - 1317006294*q^14 - 610724583*q^12 - 538477502*q^10 - 434987253*q^8 - 119464
668*q^6 + 24483303*q^4 + 17432820*q^2 + 2259900)/(2700*q^(53/2) + 24945*q^(49/2) + 739
90*q^(45/2) + 93072*q^(41/2) + 46690*q^(37/2) + 1267*q^(33/2) - 5220*q^(29/2) - 900*q^
(25/2)),
 aa_5_1_3 == 1/65536*(192*q^16 - 8536*q^14 - 24494*q^12 - 2281*q^10 - 7833*q^8 - 34170
*q^6 - 16344*q^4 - 25461*q^2 - 2673)/(4*q^(25/2) + 3*q^(21/2)),
 aa_5_1_5 == -1/65536*(1828*q^12 + 352*q^10 - 10757*q^8 + 35788*q^6 - 594*q^4 - 8100*q
^2 - 405)/(4*q^(25/2) + 3*q^(21/2)),
 aa_5_3_1 == 1/65536*(108*q^18 - 5934*q^16 + 5105*q^14 + 46195*q^12 - 51879*q^10 - 351
53*q^8 + 4283*q^6 + 7569*q^4 + 4239*q^2 + 891)/(q^(25/2) + 3*q^(21/2)),
 aa_5_3_3 == -1/65536*(5400*q^22 + 12918*q^20 - 137093*q^18 - 272731*q^16 + 357403*q^1
4 + 299725*q^12 + 100929*q^10 + 467859*q^8 + 215865*q^6 - 91935*q^4 - 169128*q^2 - 437
40)/(4*q^(37/2) + 35*q^(33/2) + 84*q^(29/2) + 45*q^(25/2)),
 aa_5_3_5 == -1/65536*(78144*q^16 + 364169*q^14 - 491094*q^12 - 1068351*q^10 + 1162018
*q^8 + 332347*q^6 - 358650*q^4 - 114885*q^2 - 4050)/(36*q^(33/2) + 215*q^(29/2) + 181*
q^(25/2) + 30*q^(21/2)),
 aa_5_5_1 == -1/65536*(180*q^22 - 360*q^20 - 1329*q^18 - 1389*q^16 - 10180*q^14 + 2507
6*q^12 + 40794*q^10 - 62542*q^8 + 3864*q^6 + 3132*q^4 + 2511*q^2 + 243)/(5*q^(29/2) +
18*q^(25/2) + 9*q^(21/2)),
 aa_5_5_3 == -1/65536*(10290*q^22 + 43395*q^20 - 173337*q^18 - 602962*q^16 + 701240*q^
14 + 1757970*q^12 - 1345366*q^10 - 1223160*q^8 + 844206*q^6 + 21555*q^4 - 31401*q^2 -
2430)/(5*q^(37/2) + 45*q^(33/2) + 113*q^(29/2) + 59*q^(25/2) - 30*q^(21/2)),
 aa_5_5_5 == -1/65536*(5415*q^16 + 32830*q^14 - 2142*q^12 - 121450*q^10 + 28240*q^8 +
135290*q^6 - 88578*q^4 + 8370*q^2 + 2025)/(3*q^(33/2) + 20*q^(29/2) + 25*q^(25/2)),
 bb_5_1_1 == -1/196608*(291600*q^34 + 2907900*q^32 + 32753160*q^30 + 149869395*q^28 -
96175800*q^26 - 1388122677*q^24 - 1816814730*q^22 - 673012947*q^20 - 2204709010*q^18 -
4183229964*q^16 - 2019366082*q^14 - 127366501*q^12 - 1024292202*q^10 - 1272297663*q^8
- 404176500*q^6 + 54476469*q^4 + 49965660*q^2 + 6779700)/(2700*q^26 + 24945*q^24 + 739
90*q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12),
 bb_5_1_3 == 1/196608*(576*q^16 + 8184*q^14 + 17094*q^12 + 22645*q^10 + 2281*q^8 + 236
58*q^6 + 7776*q^4 + 22761*q^2 + 2673)/(4*q^12 + 3*q^10),
 bb_5_1_5 == 1/196608*(60*q^12 + 1728*q^10 + 781*q^8 + 16476*q^6 - 4158*q^4 - 4212*q^2
- 243)/(4*q^12 + 3*q^10),
 bb_5_3_1 == -3/65536*(20*q^18 - 1386*q^16 + 1159*q^14 + 17385*q^12 - 1473*q^10 - 2461
1*q^8 + 13245*q^6 + 5427*q^4 + 12393*q^2 + 2673)/(q^12 + 3*q^10),
 bb_5_3_3 == 3/65536*(216*q^22 - 570*q^20 - 18933*q^18 - 71519*q^16 - 100329*q^14 - 35
063*q^12 + 38725*q^10 - 123273*q^8 - 114291*q^6 - 157707*q^4 - 125388*q^2 - 43740)/(4*
q^18 + 35*q^16 + 84*q^14 + 45*q^12),
 bb_5_3_5 == -3/65536*(320*q^16 + 1737*q^14 - 4390*q^12 - 8591*q^10 + 157370*q^8 + 252
03*q^6 + 112590*q^4 + 62451*q^2 + 2430)/(36*q^16 + 215*q^14 + 181*q^12 + 30*q^10),
 bb_5_5_1 == 5/196608*(108*q^22 + 648*q^20 - 3735*q^18 + 5853*q^16 + 84492*q^14 + 1882
8*q^12 - 256546*q^10 - 31890*q^8 + 169632*q^6 + 96228*q^4 + 37665*q^2 + 3645)/(5*q^14
+ 18*q^12 + 9*q^10),
 bb_5_5_3 == -5/196608*(342*q^22 + 1257*q^20 + 2085*q^18 - 31430*q^16 - 390368*q^14 -
350282*q^12 + 1479350*q^10 + 570136*q^8 - 1001838*q^6 - 419175*q^4 + 146205*q^2 + 1215
0)/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^12 - 30*q^10),
 bb_5_5_5 == 5/196608*(45*q^16 - 150*q^14 - 570*q^12 + 2618*q^10 - 3896*q^8 - 498*q^6
+ 25866*q^4 + 8910*q^2 + 6075)/(3*q^16 + 20*q^14 + 25*q^12),
 cc_5 == -1/16384*(12*q^16 - 176*q^14 - 681*q^12 + 201*q^10 + 279*q^8 - 978*q^6 - 279*
q^4 + 513*q^2 + 405)/q^(19/2),
 dd_5 == 0]
```

In [ ]:
```

```

In [844]:
```
#sol
```

In [845]:
```
######################          var substitution-05          ##########################
```

In [846]:
```
#from list of solve results, compare left hand sides of expressions to variable names aa
#if there was a match, assign the right hand side of solve results to A, B, C, D
#therefore aa, bb, cc and dd are intact
for ii in range(len(sol)):
    for i in range(1,6):
        if var('cc_'+str(i))==sol[ii].lhs():
            C[i]=sol[ii].rhs()
            if C[i]!=0:
                print("C",i,C[i])
        if var('dd_'+str(i))==sol[ii].lhs():
            if D[i]!=0:
                D[i]=sol[ii].rhs()
            print("D",i,D[i])
        for j in range(6):
            for m in range(6):
                if var('aa_'+str(i)+'_'+str(j)+'_'+str(m))==sol[ii].lhs():
                    A[i][j][m]=sol[ii].rhs()
                    if A[i][j][m]!=0:
                        print("A",i,j,m,A[i][j][m])
                if var('bb_'+str(i)+'_'+str(j)+'_'+str(m))==sol[ii].lhs():
                    B[i][j][m]=sol[ii].rhs()
                    if B[i][j][m]!=0:
                        print("B",i,j,m,B[i][j][m])
```

```
('A', 5, 1, 1, 1/65536*(97200*q^34 + 969300*q^32 + 10312920*q^30 + 40437585*q^28 - 9
9942680*q^26 - 718547767*q^24 - 1053139982*q^22 - 468613281*q^20 - 625150182*q^18 -
1543359636*q^16 - 1317006294*q^14 - 610724583*q^12 - 538477502*q^10 - 434987253*q^8
- 119464668*q^6 + 24483303*q^4 + 17432820*q^2 + 2259900)/(2700*q^(53/2) + 24945*q^(4
9/2) + 73990*q^(45/2) + 93072*q^(41/2) + 46690*q^(37/2) + 1267*q^(33/2) - 5220*q^(2
9/2) - 900*q^(25/2)))
('A', 5, 1, 3, 1/65536*(192*q^16 - 8536*q^14 - 24494*q^12 - 2281*q^10 - 7833*q^8 - 3
4170*q^6 - 16344*q^4 - 25461*q^2 - 2673)/(4*q^(25/2) + 3*q^(21/2)))
('A', 5, 1, 5, -1/65536*(1828*q^12 + 352*q^10 - 10757*q^8 + 35788*q^6 - 594*q^4 - 81
00*q^2 - 405)/(4*q^(25/2) + 3*q^(21/2)))
('A', 5, 3, 1, 1/65536*(108*q^18 - 5934*q^16 + 5105*q^14 + 46195*q^12 - 51879*q^10 -
35153*q^8 + 4283*q^6 + 7569*q^4 + 4239*q^2 + 891)/(q^(25/2) + 3*q^(21/2)))
('A', 5, 3, 3, -1/65536*(5400*q^22 + 12918*q^20 - 137093*q^18 - 272731*q^16 + 357403
*q^14 + 299725*q^12 + 100929*q^10 + 467859*q^8 + 215865*q^6 - 91935*q^4 - 169128*q^2
- 43740)/(4*q^(37/2) + 35*q^(33/2) + 84*q^(29/2) + 45*q^(25/2)))
('A', 5, 3, 5, -1/65536*(78144*q^16 + 364169*q^14 - 491094*q^12 - 1068351*q^10 + 116
2018*q^8 + 332347*q^6 - 358650*q^4 - 114885*q^2 - 4050)/(36*q^(33/2) + 215*q^(29/2)
+ 181*q^(25/2) + 30*q^(21/2)))
('A', 5, 5, 1, -1/65536*(180*q^22 - 360*q^20 - 1329*q^18 - 1389*q^16 - 10180*q^14 +
25076*q^12 + 40794*q^10 - 62542*q^8 + 3864*q^6 + 3132*q^4 + 2511*q^2 + 243)/(5*q^(2
9/2) + 18*q^(25/2) + 9*q^(21/2)))
('A', 5, 5, 3, -1/65536*(10290*q^22 + 43395*q^20 - 173337*q^18 - 602962*q^16 + 70124
0*q^14 + 1757970*q^12 - 1345366*q^10 - 1223160*q^8 + 844206*q^6 + 21555*q^4 - 31401*
q^2 - 2430)/(5*q^(37/2) + 45*q^(33/2) + 113*q^(29/2) + 59*q^(25/2) - 30*q^(21/2)))
('A', 5, 5, 5, -1/65536*(5415*q^16 + 32830*q^14 - 2142*q^12 - 121450*q^10 + 28240*q^
8 + 135290*q^6 - 88578*q^4 + 8370*q^2 + 2025)/(3*q^(33/2) + 20*q^(29/2) + 25*q^(25/
2)))
('B', 5, 1, 1, -1/196608*(291600*q^34 + 2907900*q^32 + 32753160*q^30 + 149869395*q^2
8 - 96175800*q^26 - 1388122677*q^24 - 1816814730*q^22 - 673012947*q^20 - 2204709010*
q^18 - 4183229964*q^16 - 2019366082*q^14 - 127366501*q^12 - 1024292202*q^10 - 127229
7663*q^8 - 404176500*q^6 + 54476469*q^4 + 49965660*q^2 + 6779700)/(2700*q^26 + 24945
*q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12))
('B', 5, 1, 3, 1/196608*(576*q^16 + 8184*q^14 + 17094*q^12 + 22645*q^10 + 2281*q^8 +
23658*q^6 + 7776*q^4 + 22761*q^2 + 2673)/(4*q^12 + 3*q^10))
('B', 5, 1, 5, 1/196608*(60*q^12 + 1728*q^10 + 781*q^8 + 16476*q^6 - 4158*q^4 - 4212
*q^2 - 243)/(4*q^12 + 3*q^10))
('B', 5, 3, 1, -3/65536*(20*q^18 - 1386*q^16 + 1159*q^14 + 17385*q^12 - 1473*q^10 -
24611*q^8 + 13245*q^6 + 5427*q^4 + 12393*q^2 + 2673)/(q^12 + 3*q^10))
('B', 5, 3, 3, 3/65536*(216*q^22 - 570*q^20 - 18933*q^18 - 71519*q^16 - 100329*q^14
- 35063*q^12 + 38725*q^10 - 123273*q^8 - 114291*q^6 - 157707*q^4 - 125388*q^2 - 4374
0)/(4*q^18 + 35*q^16 + 84*q^14 + 45*q^12))
```

```
('B', 5, 3, 5, -3/65536*(320*q^16 + 1737*q^14 - 4390*q^12 - 8591*q^10 + 157370*q^8 +
25203*q^6 + 112590*q^4 + 62451*q^2 + 2430)/(36*q^16 + 215*q^14 + 181*q^12 + 30*q^1
0))
('B', 5, 5, 1, 5/196608*(108*q^22 + 648*q^20 - 3735*q^18 + 5853*q^16 + 84492*q^14 +
18828*q^12 - 256546*q^10 - 31890*q^8 + 169632*q^6 + 96228*q^4 + 37665*q^2 + 3645)/(5
*q^14 + 18*q^12 + 9*q^10))
('B', 5, 5, 3, -5/196608*(342*q^22 + 1257*q^20 + 2085*q^18 - 31430*q^16 - 390368*q^1
4 - 350282*q^12 + 1479350*q^10 + 570136*q^8 - 1001838*q^6 - 419175*q^4 + 146205*q^2
+ 12150)/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^12 - 30*q^10))
('B', 5, 5, 5, 5/196608*(45*q^16 - 150*q^14 - 570*q^12 + 2618*q^10 - 3896*q^8 - 498*
q^6 + 25866*q^4 + 8910*q^2 + 6075)/(3*q^16 + 20*q^14 + 25*q^12))
('C', 5, -1/16384*(12*q^16 - 176*q^14 - 681*q^12 + 201*q^10 + 279*q^8 - 978*q^6 - 27
9*q^4 + 513*q^2 + 405)/q^(19/2))
('D', 5, 0)
```

In [847]:
```
##############################      verify lateral - O5          ######################
```

In [848]:
```
#eq_aa20=
expand((f_eta(0,0,5)-f_eta(pi,0,5)-2*eps)/(2*eps^5))
```

Out[848]:
```
-6075/4096*q^34/(2700*q^26 + 24945*q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 126
7*q^16 - 5220*q^14 - 900*q^12) - 242325/16384*q^32/(2700*q^26 + 24945*q^24 + 73990*q
^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12) - 1364715/8192*q^3
0/(2700*q^26 + 24945*q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*
q^14 - 900*q^12) - 49956465/65536*q^28/(2700*q^26 + 24945*q^24 + 73990*q^22 + 93072*
q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12) + 4007325/8192*q^26/(2700*q^26
+ 24945*q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^
12) + 462707559/65536*q^24/(2700*q^26 + 24945*q^24 + 73990*q^22 + 93072*q^20 + 46690
*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12) + 302802455/32768*q^22/(2700*q^26 + 24945*
q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12) - 28
5/32768*q^22/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^12 - 30*q^10) + 81/8192*q^22/(4*q^1
8 + 35*q^16 + 84*q^14 + 45*q^12) + 45/16384*q^22/(5*q^14 + 18*q^12 + 9*q^10) + 22433
7649/65536*q^20/(2700*q^26 + 24945*q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 126
7*q^16 - 5220*q^14 - 900*q^12) - 2095/65536*q^20/(5*q^18 + 45*q^16 + 113*q^14 + 59*q
^12 - 30*q^10) - 855/32768*q^20/(4*q^18 + 35*q^16 + 84*q^14 + 45*q^12) + 135/8192*q^
20/(5*q^14 + 18*q^12 + 9*q^10) + 1102354505/98304*q^18/(2700*q^26 + 24945*q^24 + 739
90*q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12) - 3475/65536*q
^18/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^12 - 30*q^10) - 56799/65536*q^18/(4*q^18 + 3
5*q^16 + 84*q^14 + 45*q^12) - 6225/65536*q^18/(5*q^14 + 18*q^12 + 9*q^10) - 15/16384
*q^18/(q^12 + 3*q^10) + 348602497/16384*q^16/(2700*q^26 + 24945*q^24 + 73990*q^22 +
93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12) + 78575/98304*q^16/(5*q^
18 + 45*q^16 + 113*q^14 + 59*q^12 - 30*q^10) - 214557/65536*q^16/(4*q^18 + 35*q^16 +
84*q^14 + 45*q^12) - 15/1024*q^16/(36*q^16 + 215*q^14 + 181*q^12 + 30*q^10) + 75/655
36*q^16/(3*q^16 + 20*q^14 + 25*q^12) + 9755/65536*q^16/(5*q^14 + 18*q^12 + 9*q^10) +
3/1024*q^16/(4*q^12 + 3*q^10) + 2079/32768*q^16/(q^12 + 3*q^10) + 1009683041/98304*q
^14/(2700*q^26 + 24945*q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 522
0*q^14 - 900*q^12) + 60995/6144*q^14/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^12 - 30*q^1
0) - 300987/65536*q^14/(4*q^18 + 35*q^16 + 84*q^14 + 45*q^12) - 5211/65536*q^14/(36*
q^16 + 215*q^14 + 181*q^12 + 30*q^10) - 125/32768*q^14/(3*q^16 + 20*q^14 + 25*q^12)
+ 35205/16384*q^14/(5*q^14 + 18*q^12 + 9*q^10) + 341/8192*q^14/(4*q^12 + 3*q^10) - 3
477/65536*q^14/(q^12 + 3*q^10) + 127366501/196608*q^12/(2700*q^26 + 24945*q^24 + 739
90*q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12) + 875705/98304
*q^12/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^12 - 30*q^10) - 105189/65536*q^12/(4*q^18
+ 35*q^16 + 84*q^14 + 45*q^12) + 6585/32768*q^12/(36*q^16 + 215*q^14 + 181*q^12 + 30
*q^10) - 475/32768*q^12/(3*q^16 + 20*q^14 + 25*q^12) + 7845/16384*q^12/(5*q^14 + 18*
q^12 + 9*q^10) + 2859/32768*q^12/(4*q^12 + 3*q^10) - 52155/65536*q^12/(q^12 + 3*q^1
0) + 170715367/32768*q^10/(2700*q^26 + 24945*q^24 + 73990*q^22 + 93072*q^20 + 46690*
q^18 + 1267*q^16 - 5220*q^14 - 900*q^12) - 3698375/98304*q^10/(5*q^18 + 45*q^16 + 11
3*q^14 + 59*q^12 - 30*q^10) + 116175/65536*q^10/(4*q^18 + 35*q^16 + 84*q^14 + 45*q^1
2) + 25773/65536*q^10/(36*q^16 + 215*q^14 + 181*q^12 + 30*q^10) + 6545/98304*q^10/(3
*q^16 + 20*q^14 + 25*q^12) - 641365/98304*q^10/(5*q^14 + 18*q^12 + 9*q^10) + 24373/1
96608*q^10/(4*q^12 + 3*q^10) + 4419/65536*q^10/(q^12 + 3*q^10) + 424099221/65536*q^
8/(2700*q^26 + 24945*q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*
q^14 - 900*q^12) - 356335/24576*q^8/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^12 - 30*q^1
0) - 369819/65536*q^8/(4*q^18 + 35*q^16 + 84*q^14 + 45*q^12) - 236055/32768*q^8/(36*
q^16 + 215*q^14 + 181*q^12 + 30*q^10) - 2435/24576*q^8/(3*q^16 + 20*q^14 + 25*q^12)
- 26575/32768*q^8/(5*q^14 + 18*q^12 + 9*q^10) + 1531/98304*q^8/(4*q^12 + 3*q^10) + 7
3833/65536*q^8/(q^12 + 3*q^10) + 33681375/16384*q^6/(2700*q^26 + 24945*q^24 + 73990*
q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12) + 834865/32768*q^
6/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^12 - 30*q^10) - 342873/65536*q^6/(4*q^18 + 35*
q^16 + 84*q^14 + 45*q^12) - 75609/65536*q^6/(36*q^16 + 215*q^14 + 181*q^12 + 30*q^1
0) - 415/32768*q^6/(3*q^16 + 20*q^14 + 25*q^12) + 8835/2048*q^6/(5*q^14 + 18*q^12 +
9*q^10) + 6689/32768*q^6/(4*q^12 + 3*q^10) - 39735/65536*q^6/(q^12 + 3*q^10) - 18158
823/65536*q^4/(2700*q^26 + 24945*q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 1267*
q^16 - 5220*q^14 - 900*q^12) + 698625/65536*q^4/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^
12 - 30*q^10) - 473121/65536*q^4/(4*q^18 + 35*q^16 + 84*q^14 + 45*q^12) - 168885/327
68*q^4/(36*q^16 + 215*q^14 + 181*q^12 + 30*q^10) + 21555/32768*q^4/(3*q^16 + 20*q^14
+ 25*q^12) + 40095/16384*q^4/(5*q^14 + 18*q^12 + 9*q^10) + 603/32768*q^4/(4*q^12 + 3
*q^10) - 16281/65536*q^4/(q^12 + 3*q^10) - 4163805/16384*q^2/(2700*q^26 + 24945*q^24
+ 73990*q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12) - 243675/
65536*q^2/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^12 - 30*q^10) - 94041/16384*q^2/(4*q^1
8 + 35*q^16 + 84*q^14 + 45*q^12) - 187353/65536*q^2/(36*q^16 + 215*q^14 + 181*q^12 +
```

```
30*q^10) + 7425/32768*q^2/(3*q^16 + 20*q^14 + 25*q^12) + 62775/65536*q^2/(5*q^14 + 1
8*q^12 + 9*q^10) + 6183/65536*q^2/(4*q^12 + 3*q^10) - 37179/65536*q^2/(q^12 + 3*q^1
0) - 564975/16384/(2700*q^26 + 24945*q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 1
267*q^16 - 5220*q^14 - 900*q^12) - 10125/32768/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^1
2 - 30*q^10) - 32805/16384/(4*q^18 + 35*q^16 + 84*q^14 + 45*q^12) - 3645/32768/(36*q
^16 + 215*q^14 + 181*q^12 + 30*q^10) + 10125/65536/(3*q^16 + 20*q^14 + 25*q^12) + 60
75/65536/(5*q^14 + 18*q^12 + 9*q^10) + 405/32768/(4*q^12 + 3*q^10) - 8019/65536/(q^1
2 + 3*q^10)
```

In [849]:
```
#eq_aa21=
expand((f_eta(0,0,5)-f_eta(0,pi,5)-2*eps)/(2*eps^5))
```

Out[849]:
```
-6075/4096*q^34/(2700*q^26 + 24945*q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 126
7*q^16 - 5220*q^14 - 900*q^12) - 242325/16384*q^32/(2700*q^26 + 24945*q^24 + 73990*q
^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12) - 1364715/8192*q^3
0/(2700*q^26 + 24945*q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*
q^14 - 900*q^12) - 49956465/65536*q^28/(2700*q^26 + 24945*q^24 + 73990*q^22 + 93072*
q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12) + 4007325/8192*q^26/(2700*q^26
+ 24945*q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^
12) + 462707559/65536*q^24/(2700*q^26 + 24945*q^24 + 73990*q^22 + 93072*q^20 + 46690
*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12) + 302802455/32768*q^22/(2700*q^26 + 24945*
q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12) - 28
5/32768*q^22/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^12 - 30*q^10) + 81/8192*q^22/(4*q^1
8 + 35*q^16 + 84*q^14 + 45*q^12) + 45/16384*q^22/(5*q^14 + 18*q^12 + 9*q^10) + 22433
7649/65536*q^20/(2700*q^26 + 24945*q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 126
7*q^16 - 5220*q^14 - 900*q^12) - 2095/65536*q^20/(5*q^18 + 45*q^16 + 113*q^14 + 59*q
^12 - 30*q^10) - 855/32768*q^20/(4*q^18 + 35*q^16 + 84*q^14 + 45*q^12) + 135/8192*q^
20/(5*q^14 + 18*q^12 + 9*q^10) + 1102354505/98304*q^18/(2700*q^26 + 24945*q^24 + 739
90*q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12) - 3475/65536*q
^18/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^12 - 30*q^10) - 56799/65536*q^18/(4*q^18 + 3
5*q^16 + 84*q^14 + 45*q^12) - 6225/65536*q^18/(5*q^14 + 18*q^12 + 9*q^10) - 15/16384
*q^18/(q^12 + 3*q^10) + 348602497/16384*q^16/(2700*q^26 + 24945*q^24 + 73990*q^22 +
93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12) + 78575/98304*q^16/(5*q^
18 + 45*q^16 + 113*q^14 + 59*q^12 - 30*q^10) - 214557/65536*q^16/(4*q^18 + 35*q^16 +
84*q^14 + 45*q^12) - 15/1024*q^16/(36*q^16 + 215*q^14 + 181*q^12 + 30*q^10) + 75/655
36*q^16/(3*q^16 + 20*q^14 + 25*q^12) + 9755/65536*q^16/(5*q^14 + 18*q^12 + 9*q^10) +
3/1024*q^16/(4*q^12 + 3*q^10) + 2079/32768*q^16/(q^12 + 3*q^10) + 1009683041/98304*q
^14/(2700*q^26 + 24945*q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 522
0*q^14 - 900*q^12) + 60995/6144*q^14/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^12 - 30*q^1
0) - 300987/65536*q^14/(4*q^18 + 35*q^16 + 84*q^14 + 45*q^12) - 5211/65536*q^14/(36*
q^16 + 215*q^14 + 181*q^12 + 30*q^10) - 125/32768*q^14/(3*q^16 + 20*q^14 + 25*q^12)
+ 35205/16384*q^14/(5*q^14 + 18*q^12 + 9*q^10) + 341/8192*q^14/(4*q^12 + 3*q^10) - 3
477/65536*q^14/(q^12 + 3*q^10) + 127366501/196608*q^12/(2700*q^26 + 24945*q^24 + 739
90*q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12) + 875705/98304
*q^12/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^12 - 30*q^10) - 105189/65536*q^12/(4*q^18
+ 35*q^16 + 84*q^14 + 45*q^12) + 6585/32768*q^12/(36*q^16 + 215*q^14 + 181*q^12 + 30
*q^10) - 475/32768*q^12/(3*q^16 + 20*q^14 + 25*q^12) + 7845/16384*q^12/(5*q^14 + 18*
q^12 + 9*q^10) + 2859/32768*q^12/(4*q^12 + 3*q^10) - 52155/65536*q^12/(q^12 + 3*q^1
0) + 170715367/32768*q^10/(2700*q^26 + 24945*q^24 + 73990*q^22 + 93072*q^20 + 46690*
q^18 + 1267*q^16 - 5220*q^14 - 900*q^12) - 3698375/98304*q^10/(5*q^18 + 45*q^16 + 11
3*q^14 + 59*q^12 - 30*q^10) + 116175/65536*q^10/(4*q^18 + 35*q^16 + 84*q^14 + 45*q^1
2) + 25773/65536*q^10/(36*q^16 + 215*q^14 + 181*q^12 + 30*q^10) + 6545/98304*q^10/(3
*q^16 + 20*q^14 + 25*q^12) - 641365/98304*q^10/(5*q^14 + 18*q^12 + 9*q^10) + 24373/1
96608*q^10/(4*q^12 + 3*q^10) + 4419/65536*q^10/(q^12 + 3*q^10) + 424099221/65536*q^
8/(2700*q^26 + 24945*q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*
q^14 - 900*q^12) - 356335/24576*q^8/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^12 - 30*q^1
0) - 369819/65536*q^8/(4*q^18 + 35*q^16 + 84*q^14 + 45*q^12) - 236055/32768*q^8/(36*
q^16 + 215*q^14 + 181*q^12 + 30*q^10) - 2435/24576*q^8/(3*q^16 + 20*q^14 + 25*q^12)
- 26575/32768*q^8/(5*q^14 + 18*q^12 + 9*q^10) + 1531/98304*q^8/(4*q^12 + 3*q^10) + 7
3833/65536*q^8/(q^12 + 3*q^10) + 33681375/16384*q^6/(2700*q^26 + 24945*q^24 + 73990*
q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12) + 834865/32768*q^
6/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^12 - 30*q^10) - 342873/65536*q^6/(4*q^18 + 35*
q^16 + 84*q^14 + 45*q^12) - 75609/65536*q^6/(36*q^16 + 215*q^14 + 181*q^12 + 30*q^1
0) - 415/32768*q^6/(3*q^16 + 20*q^14 + 25*q^12) + 8835/2048*q^6/(5*q^14 + 18*q^12 +
9*q^10) + 6689/32768*q^6/(4*q^12 + 3*q^10) - 39735/65536*q^6/(q^12 + 3*q^10) - 18158
823/65536*q^4/(2700*q^26 + 24945*q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 1267*
q^16 - 5220*q^14 - 900*q^12) + 698625/65536*q^4/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^
12 - 30*q^10) - 473121/65536*q^4/(4*q^18 + 35*q^16 + 84*q^14 + 45*q^12) - 168885/327
68*q^4/(36*q^16 + 215*q^14 + 181*q^12 + 30*q^10) + 21555/32768*q^4/(3*q^16 + 20*q^14
+ 25*q^12) + 40095/16384*q^4/(5*q^14 + 18*q^12 + 9*q^10) + 603/32768*q^4/(4*q^12 + 3
*q^10) - 16281/65536*q^4/(q^12 + 3*q^10) - 4163805/16384*q^2/(2700*q^26 + 24945*q^24
+ 73990*q^22 + 93072*q^20 + 46690*q^18 + 1267*q^16 - 5220*q^14 - 900*q^12) - 243675/
65536*q^2/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^12 - 30*q^10) - 94041/16384*q^2/(4*q^1
8 + 35*q^16 + 84*q^14 + 45*q^12) - 187353/65536*q^2/(36*q^16 + 215*q^14 + 181*q^12 +
```

```
30*q^10) + 7425/32768*q^2/(3*q^16 + 20*q^14 + 25*q^12) + 62775/65536*q^2/(5*q^14 + 1
8*q^12 + 9*q^10) + 6183/65536*q^2/(4*q^12 + 3*q^10) - 37179/65536*q^2/(q^12 + 3*q^1
0) - 564975/16384/(2700*q^26 + 24945*q^24 + 73990*q^22 + 93072*q^20 + 46690*q^18 + 1
267*q^16 - 5220*q^14 - 900*q^12) - 10125/32768/(5*q^18 + 45*q^16 + 113*q^14 + 59*q^1
2 - 30*q^10) - 32805/16384/(4*q^18 + 35*q^16 + 84*q^14 + 45*q^12) - 3645/32768/(36*q
^16 + 215*q^14 + 181*q^12 + 30*q^10) + 10125/65536/(3*q^16 + 20*q^14 + 25*q^12) + 60
75/65536/(5*q^14 + 18*q^12 + 9*q^10) + 405/32768/(4*q^12 + 3*q^10) - 8019/65536/(q^1
2 + 3*q^10)
```

In [850]:
```
################################### verify kinematic BC - o5 ##########################
```

In [851]:
```
expr=(f_eta_t(x,t,5)+f_u(x,t,4)*f_eta_x(x,t,4)-f_w(x,t,5)).simplify_full().coefficient(e
```

In [852]:
```
################################### verify dynamic BC - o5 ##########################
```

In [853]:
```
(f_phi_t(x,t,5,4,4,5)+1/2*(f_u(x,t,4,3,3)^2+f_w(x,t,4,3,3)^2)+f_eta(x,t,5)-fB(5)).maxima
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/opt/sagemath-8.1/local/lib/python2.7/site-packages/sage/all_cmdline.pyc in <module>()
----> 1 (f_phi_t(x,t,Integer(5),Integer(4),Integer(4),Integer(5))+Integer(1)/Integer(2
)*(f_u(x,t,Integer(4),Integer(3),Integer(3))**Integer(2)+f_w(x,t,Integer(4),Integer(3)
,Integer(3))**Integer(2))+f_eta(x,t,Integer(5))-fB(Integer(5))).maxima_methods().divid
e(eps**Integer(6))[Integer(1)].simplify_full()

TypeError: f_phi_t() takes exactly 3 arguments (6 given)
```