Non-convex

```
def tanhsinh(FL, FR, a, b, N=4, eps=2^-13):
    """
    Returns lower and upper bounds on the integral of a convex function F on an interval
[a+D,b-D].

    FL(x)=F(a+(b-a)*x)
    FR(x)=F(b-(b-a)*x)
    """
    X = [RIF(1/(2*exp(sinh(t)*pi/2)*cosh(sinh(t)*pi/2))) for t in srange(0,N,eps)]    #X is
a mesh on (0,1/2]
    n = len(X)
    Xmid = [(X[j]+X[j+1])/2 for j in range(n-1)]
    W = [X[j]-X[j+1] for j in range(n-1)]                                              #size
of j-th interval
    D = (b-a)*X[n-1]                                                                   #size
of first and last intervals

    YL = [FL(x) for x in X]
    YR = [FR(x) for x in X]
    U = (b-a)*sum([W[j]*(YL[j]+YL[j+1]+YR[j]+YR[j+1])/2 for j in range(n-1)])

    YLmid = [FL(x) for x in Xmid]
    YRmid = [FR(x) for x in Xmid]
    L = (b-a)*sum([W[j]*(YLmid[j]+YRmid[j]) for j in range(n-1)])

    return [L, U, D]
```

```
def doubleexp(F, N=4, eps=2^-13):
    """
    Returns lower and upper bounds on the integral of a convex function F on the interval
(0,Infinity).
    """
    X = [RIF(exp((pi/2)*sinh(t))) for t in srange(-N,N,eps)]    #X is a mesh on
(0,Infinity)
    n = len(X)
    first = X[0]
    last = X[n-1]

    Xmid = [(X[j]+X[j+1])/2 for j in range(n-1)]
    W = [X[j+1]-X[j] for j in range(n-1)]

    Y = [F(x) for x in X]
    U = sum([W[j]*(Y[j]+Y[j+1])/2 for j in range(n-1)])

    Ymid = [F(x) for x in Xmid]
    L = sum([W[j]*Ymid[j] for j in range(n-1)])

    return [L, U, first, last]
```

```
I = range(12)
```

```
K = RIF(5.27110734472)

# Estimate for |f(0)-f(1)|

def FL(x):
    return 1/sqrt(x*(1-x^2)*(K^2-x^2))
def FR(x):
```

```
    return 1/sqrt(x*(1-x)*(2-x)*(K^2-1+2*x-x^2))

[L,U,D]=tanhsinh(FL,FR,0,1)
U = U + 2*sqrt(D/((1-D^2)*(K^2-D^2))) + 2*sqrt(D/((2-D)*(1-D)*(K^2-1)))

I[0]=RIF(L.lower(),U.upper())
print I[0].endpoints()

# Estimate for |f(1)-f(k)|

def FL(x):
    return 1/sqrt((1+(K-1)*x)*((K-1)*x)*(2+(K-1)*x)*((K-1)*(1-x))*(K+1+(K-1)*x))
def FR(x):
    return 1/sqrt((K-(K-1)*x)*((K-1)*(1-x))*(K+1-(K-1)*x)*((K-1)*x)*(2*K-(K-1)*x))

[L,U,D]=tanhsinh(FL,FR,1,K)
U = U + sqrt(2*D/(K^2-(1+D)^2)) + 2*sqrt(D/((K-D)*(2*K-D)*((K-D)^2-1)))

I[1]=RIF(L.lower(),U.upper())
print I[1].endpoints()
```

```
    (0.500482492919239, 0.500482504322058)
    (0.400385993317482, 0.400386005493315)
```

```
# Estimate for |g(0)-g(1)|

def FL(x):
    return 1/sqrt((1-x^2)*(K^2-x^2))
def FR(x):
    return 1/sqrt(x*(2-x)*(K^2-1+2*x-x^2))

[L,U,D] = tanhsinh(FL,FR,0,1)
U = U+2*sqrt(D/((2-D)*(1-D)*(K^2-1)))

I[2]=RIF(L.lower(),U.upper())
print I[2].endpoints()

# Estimate for |g(1)-g(k)|

def FL(x):
    return 1/sqrt(((K-1)*x)*(2+(K-1)*x)*((K-1)*(1-x))*(K+1+(K-1)*x))
def FR(x):
    return 1/sqrt(((K-1)*(1-x))*((K+1)-(K-1)*x)*((K-1)*x)*(2*K-(K-1)*x))

[L,U,D] = tanhsinh(FL,FR,1,K)
U = U + sqrt(2*D/(K^2-(1+D)^2)) + 2*sqrt(D/((2*K-D)*((K-D)^2-1)))

I[3]=RIF(L.lower(),U.upper())
print I[3].endpoints()
```

```
    (0.300738235179851, 0.300738239979475)
    (0.581911579444022, 0.581911593792079)
```

```
EL_bef = 2*I[3]/I[2]
print EL_bef.endpoints()
```

```
    (3.86988751070524, 3.86988766788550)
```

```
k = 5.27110734472

F(x)=1/sqrt(abs(x*(x^2-1)*(x^2-k^2)))
G(x)=1/sqrt(abs((x^2-1)*(x^2-k^2)))

print F.nintegral(x,0,1)[0]
print F.nintegral(x,1,k)[0]
J2 = G.nintegral(x,0,1)[0]
J3 = G.nintegral(x,1,k)[0]
print J2
```

```
print J3
print 2*J3/J2
```

```
0.500482496721
0.400385997377
0.30073823678
0.581911584228
3.86988758368
```

```
Z = map(RIF,[-3.33297982345, -0.26873921366, 0, 1, 2.94288195633])

# Estimate for |phi(z_m)-phi(z_{m+1})|

for m in range(4):
    def FL(x):
        return 1/sqrt(abs(prod([Z[m]-Z[j]+(Z[m+1]-Z[m])*x for j in range(5)])))
    def FR(x):
        return 1/sqrt(abs(prod([Z[m+1]-Z[j]-(Z[m+1]-Z[m])*x for j in range(5)])))

    [L,U,D] = tanhsinh(FL,FR,Z[m],Z[m+1])
    U = U + 2*sqrt(D/(prod([Z[m]-Z[j] for j in range(5) if j < m])*prod([Z[j]-(Z[m]+D) for
j in range(5) if j > m]))) + \
            2*sqrt(D/(prod([(Z[m+1]-D)-Z[j] for j in range(5) if j < m+1])*prod([Z[j]-
Z[m+1] for j in range(5) if j > m+1])))

    I[m+4] = RIF(L.lower(),U.upper())
    print I[m+4].endpoints()
```

```
(1.03682340557612, 1.03682344398208)
(0.943128409696277, 0.943128430639702)
(1.29602925190202, 1.29602928458305)
(0.754502722746341, 0.754502742640079)
```

```
P = RIF(0.17317940636)

# Estimate for |psi(z_{m})-psi(z_{m+1})| for m = 0, 1, 3

for m in range(4):
    if m!=2:
        def FL(x):
            return sqrt(abs((Z[m]-P+(Z[m+1]-Z[m])*x)/prod([Z[m]-Z[j]+(Z[m+1]-Z[m])*x for j
in range(5)])))
        def FR(x):
            return sqrt(abs((Z[m+1]-P-(Z[m+1]-Z[m])*x)/prod([Z[m+1]-Z[j]-(Z[m+1]-Z[m])*x
for j in range(5)])))

        [L,U,D] = tanhsinh(FL,FR,Z[m],Z[m+1])
        bool = (m < 2)
        U = U + 2*sqrt(D*abs(P-(Z[m]+D*(1-bool)))/(prod([Z[m]-Z[j] for j in range(5) if j <
m])*prod([Z[j]-(Z[m]+D) for j in range(5) if j > m]))) + 2*sqrt(D*abs(P-(Z[m+1]-
D*bool))/(prod([(Z[m+1]-D)-Z[j] for j in range(5) if j < m+1])*prod([Z[j]-Z[m+1] for j in
range(5) if j > m+1])))

        I[m-(1-bool)+8] = RIF(L.lower(),U.upper())
        print I[m-(1-bool)+8].endpoints()
```

```
(1.06895514575184, 1.06895517538449)
(0.512964353079769, 0.512964364187398)
(0.908877581965566, 0.908877603158245)
```

```
# Estimate for |psi(z_4)-\psi(\infty)|

def F(x):
    return sqrt(abs((Z[4]-P+x)/prod([Z[4]-Z[j]+x for j in range(5)])))

[L1,U1,first,last] = doubleexp(F)
U1 = U1 + 2*sqrt(first*(Z[4]+first-P)/prod([Z[4]-Z[j] for j in range(4)])) +
sqrt(RIF(2))/last
```

```
# Estimate for |psi(\infty)-\psi(z_0)|

def F(x):
    return sqrt(abs((Z[0]-P-x)/prod([Z[0]-Z[j]-x for j in range(5)])))

[L2,U2,first,last] = doubleexp(F)
U2 = U2 + 2*sqrt(first*(P-(Z[0]-first))/prod([Z[j]-Z[0] for j in range(5) if j>0])) +
sqrt(RIF(2))/last

# Estimate for |psi(z_4)-\psi(z_0)|

I[11] = RIF((L1+L2).lower(),(U1+U2).upper())
print I[11].endpoints()
```

```
(1.02592870063152, 1.02592873889047)
```

```
EL_aft = RIF((I[8]+I[10])^2/(I[8]*I[9]+I[10]*(I[11]-I[9])),I[8]/I[9]+I[10]/(I[11]-I[9]))
print EL_aft.endpoints()

d = 1
b = I[0]/I[1]-1
a = I[4]/I[7]
c = b*a

D = 1
C = (I[6]-I[4])/I[7]
B = I[5]/I[7]-1
A = I[4]/I[7]
M=max([b/B, B/b, c/C, C/c])
print M.upper()

EL_cor=RIF(EL_aft/M,M*EL_aft)
print EL_cor.endpoints()
```

```
(3.85569208440585, 3.85569249820843)
1.00000035775881
(3.85569070499854, 3.85569387761638)
```

```
Z = [-3.33297982345, -0.26873921366, 0, 1, 2.94288195633]
P = 0.17317940636

F(x) = 1/sqrt(abs(prod([x-Z[j] for j in range(5)])))
G(x) = sqrt(abs((x-P)/prod([x-Z[j] for j in range(5)])))

for j in range(4):
    print F.nintegral(x,Z[j],Z[j+1])[0]

J8 = G.nintegral(x,Z[0],Z[1])[0]
J9 = G.nintegral(x,Z[1],Z[2])[0]
J10 = G.nintegral(x,Z[3],Z[4])[0]
J11 = numerical_integral(G,-Infinity,Z[0])[0] + numerical_integral(G,Z[4],+Infinity)[0]
print J8
print J9
print J10
print J11
print J8/J9+(J11/J9-1)*J10/J9
```

```
1.03682341838
0.943128416679
1.2960292628
0.754502729379
1.06895515563
0.512964356783
0.908877589032
1.02592871356
3.85569234685
```