We use Online LWE Estimator

(https://lwe-estimator.readthedocs.io/en/latest/index.html)

load("https://bitbucket.org/malb/lwe-estimator/raw/HEAD/estimator.py")

#######Comfort parameter; 128bit###############

n=512

q=256

p=64

k=1.

sttdev=sqrt(k/2)

hs=128

hr=128

R\_alpha=sqrt(3.141592653589793238462643383/6.)/p

alpha=alphaf(sigmaf(sttdev),q)

Qcost=partial(BKZ.ADPS16, mode="quantum")

Ccost=partial(BKZ.ADPS16, mode="classical")

Pcost=partial(BKZ.ADPS16, mode="paranoid")

primald=partial(drop\_and\_solve, primal\_usvp)

duald=partial(drop\_and\_solve, dual\_scale)

print("###############Comfort parameter#####################")

print("Each First row is LWE // Second row is LWR")

print("Q\_dual")

print(duald(n, alpha, q, secret\_distribution=((-1,1),hs), reduction\_cost\_model=Qcost))

print(duald(n, R\_alpha, q, secret\_distribution=((-1,1),hr), reduction\_cost\_model=Qcost))

print("C\_dual")

print(duald(n, alpha, q, secret\_distribution=((-1,1),hs), reduction\_cost\_model=Ccost))

print(duald(n, R\_alpha, q, secret\_distribution=((-1,1),hr), reduction\_cost\_model=Ccost))

print("P\_dual")

print(duald(n, alpha, q, secret\_distribution=((-1,1),hs), reduction\_cost\_model=Pcost))

print(duald(n, R\_alpha, q, secret\_distribution=((-1,1),hr), reduction\_cost\_model=Pcost))

print

print("Q\_primal")

print(primald(n, alpha, q, secret\_distribution=((-1,1),hs), m=n, rotations=True, postprocess=False, reduction\_cost\_model=Qcost, decision=True))

print(primald(n, R\_alpha, q, secret\_distribution=((-1,1),hr), m=n, rotations=True, postprocess=False, reduction\_cost\_model=Qcost, decision=True))

print("C\_primal")

print(primald(n, alpha, q, secret\_distribution=((-1,1),hs), m=n, rotations=True, postprocess=False, reduction\_cost\_model=Ccost, decision=True))

print(primald(n, R\_alpha, q, secret\_distribution=((-1,1),hr), m=n, rotations=True, postprocess=False, reduction\_cost\_model=Ccost, decision=True))

print("P\_primal")

print(primald(n, alpha, q, secret\_distribution=((-1,1),hs), m=n, rotations=True, postprocess=False, reduction\_cost\_model=Pcost, decision=True))

print(primald(n, R\_alpha, q, secret\_distribution=((-1,1),hr), m=n, rotations=True, postprocess=False, reduction\_cost\_model=Pcost, decision=True))

print

#######Strong parameter; 256bit###############

#q, p, k, hs, hr is same

n=1024

print("###############Strong parameter#####################")

print("Each First row is LWE // Second row is LWR")

print("Q\_dual")

print(duald(n, alpha, q, secret\_distribution=((-1,1),hs), reduction\_cost\_model=Qcost))

print(duald(n, R\_alpha, q, secret\_distribution=((-1,1),hr), reduction\_cost\_model=Qcost))

print("C\_dual")

print(duald(n, alpha, q, secret\_distribution=((-1,1),hs), reduction\_cost\_model=Ccost))

print(duald(n, R\_alpha, q, secret\_distribution=((-1,1),hr), reduction\_cost\_model=Ccost))

print("P\_dual")

print(duald(n, alpha, q, secret\_distribution=((-1,1),hs), reduction\_cost\_model=Pcost))

print(duald(n, R\_alpha, q, secret\_distribution=((-1,1),hr), reduction\_cost\_model=Pcost))

print

print("Q\_primal")

print(primald(n, alpha, q, secret\_distribution=((-1,1),hs), m=n, rotations=True, postprocess=False, reduction\_cost\_model=Qcost, decision=True))

print(primald(n, R\_alpha, q, secret\_distribution=((-1,1),hr), m=n, rotations=True, postprocess=False, reduction\_cost\_model=Qcost, decision=True))

print("C\_primal")

print(primald(n, alpha, q, secret\_distribution=((-1,1),hs), m=n, rotations=True, postprocess=False, reduction\_cost\_model=Ccost, decision=True))

print(primald(n, R\_alpha, q, secret\_distribution=((-1,1),hr), m=n, rotations=True, postprocess=False, reduction\_cost\_model=Ccost, decision=True))

print("P\_primal")

print(primald(n, alpha, q, secret\_distribution=((-1,1),hs), m=n, rotations=True, postprocess=False, reduction\_cost\_model=Pcost, decision=True))

print(primald(n, R\_alpha, q, secret\_distribution=((-1,1),hr), m=n, rotations=True, postprocess=False, reduction\_cost\_model=Pcost, decision=True))

print